Physics-Based Character Animation

An Introduction To Realistic Multibody Dynamics

by

Liya Ye Hu

Project Supervisor: Dr. Minsi Chen

Faculty of Computing and Mathematics

Contents

Chapter 1

Introduction

Physically-based animation intends to simulate the physically plausible behaviour of virtual objects in an environment where the objects interact with each other and with external forces, such as gravity, pressure, etc.

Recently, it has caught the attention of many computer games programmers and researchers. In computer games, physics-based animation has been used in the past for creating outstandingly realistic deaths with the use of rag-dolls, or even simulations of cloth and hair. Key-framing is expensive in terms of modelling, as in the past, the artist would have to design every single frame. With physics-based animation, this work can be done by applying the physical laws of the real world. Moreover, unfeasible but realistic looking motions can be computed from feasible input motion data. Other applications can be in the field of robotics, real-time simulations of medical surgeries, virtual reality, etc.

Realistic human motion or, in fact, any realistic simulation of an environment which involves some kind of interaction with external forces, has been a subject of interest recently. Its applications open up to a wide range of possibilities such as computer games productions, real-time simulations for scientific research, real-time control of a character, virtual reality or even in motion sensing input devices such as the *Microsoft Kinect* or *Leap Motion* [1]. Many different approaches have been attempted previously to approximate the simulations to reality. These approaches can be classified in three main blocks:

- *Fully Physical* Solutions that fall in this classification require no input animation data and they are entirely based on the laws of physics. Therefore, some knowledge of body dynamics, biomechanics, numerical integration and optimization algorithms is required. Some fully physical solutions are [43][30][22].

  The main downsides of this approach are that these solutions tend to be susceptible to failure under large disturbances and their computational costs are much higher compared to those of fully kinematic approaches.

- *Fully Kinematic* These solutions reassemble and procedurally modify animations to accomplish a desired task. That is, they predefine the actions of the models involved in the animation and therefore, there is little space for unexpected interactions or behaviours. Some examples of fully kinematic solutions are [16][10].

  Although they are much more efficient in terms of computation, they are limited in their ability to respond to perturbations and they generally require a number of sample clips or input data that allows to tune the parameters involved in order to reach the desired level of realism. Otherwise, it would be in risk of suffering from a very stiff and marionette-like appearance. Finding the right values for tuning is time consuming, as it involves a great amount of try-and-fail. At the same time, input data and real human motion samples take an effort to collect and to analyse, not to mention that there can easily be a vast amount of data required in order to take all kinds of situations into account.

- *Hybrid*

  As can be deduced from the name, these alternative approaches are a combination of the two mentioned above: they selectively apply simulation either to a subset of the character's joints or during specific time intervals. Or by adding additional kinematic constraints to the equations of motion of the dynamic system. Several approaches that fall into this category are [4][25][27].

  The advantage of this approach is that it can take the best of both worlds: usage of sample data that can be derived to compute new realistic motions and physics-based animations in those situations for which there are no samples or to define the character's behaviour when encountering an unexpected interaction. Also, non-physical forces can be added in order to create realistic motion that does not have to be actually plausible in real life.

1.1 Project Rationale

Deciding which method to implement to create a simulation of a character can be a very difficult task for a programmer that is newly introduced to the field. This dissertation partly intends to be an introduction to physics-based character animation as we will introduce the reader to some concepts that are relevant for understanding the implementation. Such concepts are classified in various fields that are not programming related but related to physics, biomechanics, numerical integration, etc.

We intend to clarify some of the doubts that can surface when getting started and try to classify some of the existing methods in order to give a guideline to the decision making of choosing between one method or another.

Moreover, our approach is an attempt to simplify the problem and use the minimum amount of resources needed in order to get some realistic results with a low cost in implementation time and computation. It attempts to give a definition of a model that can be subject of a forward dynamics simulation using one of the most popular methods by Featherstone [15].

1.2 Project Aim and Objectives

The aim of this dissertation is to prove the difficulty of implementing a realistic animation of a character and why it still remains an open problem.

In this research our main goals is to create a physics-based character whose movements look natural enough when it interacts with external forces. In order to achieve this, firstly, we are going to construct a body with Bullet Physics, which is a physics engine for implementing physics-based environments based on C++. Secondly, the set of constraints and optimization functions have to be defined in order to achieve a realistic usage of the limbs and joints. Afterwards, an optimization controller has to be implemented, altogether with a set of constraints, resulting in the torques that will be used for the forward dynamics that should be attached to the physics engine and applied to the model.

Chapter 2

Literature Review

In this section, several approaches that belong to the different classifications described in the introduction will be discussed. The following approaches are the most popular solutions at the moment, but there are other existing proposals and some are even a combination of the ones introduced, which will not be discussed in this research.

2.1 Data-driven Simulation

Data-driven simulations rely on the collection of samples of real motion. In order to collect these set of images and process them as data, there are several methods on the market. Some examples can be found in [19], where they use particle filters constrained by human biomechanics to help identify the different parts of the human body.

This data can be used as a guideline for the animation where the model simulates the same exact poses and actions by tracking the trajectories of the motion data. The tracking is combined with a number of controllers that enable the character to actually interact with the environment and react to the different unpredicted impacts [20], maintain balance or simply react in a natural-looking manner [21].

Another approach that falls in this category is to use the captured data to generate new animations derived from the poses of the input [2][4][7][8][9][25][27]. The latter can be interesting because the motion does not have to be physically feasible but to appear realistic.

Other attempts to solve the problem divide the human motion in various transitions in between some predefined states. An example of this is the research on *Composable Controllers* [16]. They define a set of preconditions for each state, which will enable them to identify the state of the body and evaluate the possible different states they can achieve from that point of the simulation. This approach relies on biomechanics during the process of defining the preconditions for each state and the definitions of the states themselves and introduces controllers to compute the transitions in between. The transitions are based on example clips.

The concept of data-driven simulations is very intuitive and one would think that it provides very good results in terms of realism, as it actually does take input from the real world. Let us see the downsides of this approach: Data-driven character animation can be very costly both in terms of space and time, as there are many aspects to take into account. Moreover, in order to generate all of this data, a great number of tests has to be performed before-hand, which moreover sometimes it can only provide estimations and not real values.

Motion data is not applicable to models that do not have the same properties as the expected motion would not match its behaviour. Another fact is that the subtle details such as motion style are very difficult to capture and they are relevant in order to make an animation appear realistic. Thus, taking all of this into account, this approach is not ideal, but it can be a good source for estimating parameters and combining it with dynamics [4].

2.2 Biomechanics-based Simulation

Many approaches are based on constraints defined by Human biomechanics by generating *de novo*, that is without using preexisting data. biomechanics study the different existing human motions and activities by measuring the anthropometric parameters, which describe the physical properties of the body. In order to get this data, we have to proceed to track the human movements with cameras and infra-red strobes, which will provide the input. However, ones individual movements can differ from others in terms of joint movement constraints, mass distribution and therefore a different momentum of inertia. Motor redundancy can be another aspect to be taken into account, as it states that for any task that the human body can perform, there is a wide range of sequences of actions that can achieve the task. There are three types of motor redundancy:

- *Kinematic Redundancy* For a desired position of the end-effector, there are many configurations of the joints that can reach it.
- *Muscle Redundancy* The same amount of torque can be generated by many different relative contributions of the individual muscles.
- *Motor Unit* The same amount of force can be generated by many different relative contributions of the groups of motor units that coordinate the contractions of the muscles.

All of these differences affect the realism of the animation, as the construction of the body lets the observer assume a certain behaviour and we would need an inconceivable great amount of input data.

One might think that the any movement made in order to achieve a task is optimized in terms of energy and so this could simplify the equations of movements but that is not the case in real motion. For example, a motion that is computed based only on energy efficiency can induce to a low number of joints actually contributing to the movement and causing them to be in an extremely uncomfortable position, therefore achieving a very unrealistic body posture.

The various tests that can be performed in order to get the input data enable us to estimate other quantities that might be useful.

- *Virtual representation of the skeletal system in motion*
- *Joint kinematics*
- *Joint kinetics*
- *Joint energetics*

Moreover, biomechanics is widely used for defining constraints for human posture and transitions in between the different postures in order to perform a variety of actions [24]. It can also be used in order to prioritize the usage of some muscles in front of others, define different styles or analyze the stress that is caused by certain poses and therefore avoid unrealistic-looking motions [32][14].

2.3 Kinematics Based Animation

In combination with constraints and controllers, the motion of a character can be studied without taking forces into account. The movement is imposed by the constraints and controllers that compute the necessary values in order to reach the goal. These constraints can include the position of the projection of the center of mass [34], the size of the footsteps in a walking motion [9], the momentum of inertia about the center of mass, etc. Many of these approaches are combined with bio-mechanic based constraints or with input motion data [25][26] in order to produce more realistic animations. Kinematics are highly used in the field of robotics [17], although the constraints are usually more relaxed compared to those defined in an animation; they don't seek for realism as long as the goal is reached. But artificial intelligence that imitates human expressions and motion styles is catching the attention of the robotics business nowadays.

2.3.1 Denavit-Hartenberg Parameters

For a better understanding of the following algorithms used in kinematic methods, we need to introduce the *Denavit-Hartenberg* convention. *Denavit-Hartenberg* is a convention used to attach the reference frames of each joint of the system by describing the relative position between one another. This convention is the most efficient as it only uses four parameters for describing each reference frame. The convention has to be applied following this rules:

for all $j \in Joints$ **do**
  for all $DOFi \in j$ **do**
    Z_i = Rotational_Axis;
    $X_i = Z_{i-1} \times Z_i$;
    Y_i = Compute_Y_Right_handed_Coor_System;
    Compute_Parameters(i);
  end for
end for

Where *Compute_Parameters(i)* calculates the Denavit-Hartenberg (DH) parameters for the DOF i of the joint j as stated below:

- d**:** Offset along the previous z ti the common normal.
- θ**:** Angle about previous z, from old x to new x.
- a**:** Length of the common normal
- α**:** Angle about the common normal, from old z axis to new z axis.

2.3.2 Forward Kinematics:

Forward Dynamics are used for computing the position of end-effectors, which are the ends of the limbs (hands and feet in our model), given the *DH* parameters that describe the relative position of the different joints. This problem has a straightforward solution and can be defined as

$$P = TP_0$$

where P is the resulting position of the end-effector, T is a transformation matrix that specifies the *DH* parameters and P' is the original position of the end-effector. In order to compute the transformation matrices that are multiplied to the original position of the joints resulting in the positions of the different joints, two kinds of matrices are used:

The *rotation matrices* are defined as the following figure shows:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad \left[\begin{array}{ccc|c} \cos(\alpha) & 0 & \sin(\alpha) & 0 \\ 0 & 1 & 0 & 0 \\ -\sin(\alpha) & 0 & \cos(\alpha) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad \left[\begin{array}{ccc|c} \cos(\alpha) & -\sin(\alpha) & 0 & 0 \\ \sin(\alpha) & \cos(\alpha) & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Rot. of α about the x axis | Rot. of α about the y axis | Rot. of α about the z axis

Figure 2.1: Rotation matrices

The *transformation matrices* are defined as the following figure shows:

$$\left[\begin{array}{ccc|c} 1 & 0 & 0 & a \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & c \\ 0 & 0 & 1 & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \quad \left[\begin{array}{ccc|c} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d \\ \hline 0 & 0 & 0 & 1 \end{array}\right]$$

Trans. of a along the x axis | Trans. of c along the y axis | Trans. of d along the z axis

Figure 2.2: Rotation matrices

As the DH convention is defined around the parameters d, θ, a and α, the *transformation matrix*, also called the *Denavit-Hartenberg matrix*, in between links is

$$\begin{aligned} {}^{i-1}_{i}T &= \left[\begin{array}{ccc|c} 1 & 0 & 0 & a \\ 0 & \cos(\alpha) & -\sin(\alpha) & 0 \\ 0 & \sin(\alpha) & \cos(\alpha) & 0 \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \left[\begin{array}{ccc|c} \cos(\theta) & -\sin(\theta) & 0 & 0 \\ \sin(\theta) & \cos(\theta) & 0 & 0 \\ 0 & 0 & 1 & d \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \\ &= \left[\begin{array}{ccc|c} \cos(\theta) & -\sin(\theta)\cos(\alpha) & \sin(\theta)\sin(\alpha) & a\cos(\theta) \\ \sin(\theta) & \cos(\theta)\cos(\alpha) & -\cos(\theta)\sin(\alpha) & a\sin(\theta) \\ 0 & \sin(\alpha) & \cos(\alpha) & d \\ \hline 0 & 0 & 0 & 1 \end{array}\right] \end{aligned}$$

2.3.3 Inverse Kinematics:

In order to compute the change in the joint parameters to reach the desired end-effector position we apply inverse kinematics. Unlike dynamics, the forces are not taken into account and thus, the whole animation is restricted to run under predefined conditions, no unexpected interactions. The process of specifying the motion of the character under this conditions is called *motion planning*.

There are various algorithms for solving the inverse kinematic problem. One of them is to apply the *Jacobian* iteratively, in order to get an approximate solution to the equation. More specifically, this technique is called the *Jacobian inverse technique*, which is defined as follows:

Let $p_0 = p(x_0)$ and $p = p(x_0 + \Delta x)$ be the initial and desired positions of the end-effector, respectively.

The Jacobian measures the sensitivity of the values involved in the final position to the changes of the parameters. Let us have n degrees of freedom. The idea behind the Jacobian is that it can be built for a specific set of conditions or constraints of the motion, e.g. maximum angle of a DoF, maximum height of the joint, etc. That is, an homogeneous matrix containing the information regarding the conditions that we want to fulfill. Therefore, the Jacobian will be a $3 \times n$ matrix, where each element of the Jacobian is

$$\frac{\partial X_i}{\partial \theta_j}$$

for $i = 1:3$ and $j = 1:n$. The inverse kinematics problem can be writen as an equation as follows

$$\Delta x = J \Delta \theta$$

where J is the Jacobian and $\Delta\theta$ is the change in the parameters of each joint, which is what we want to compute. Solving this equation is impracticable, so in order to get the solution, the method converts the equation above into

$$\begin{aligned} \Delta\theta &= J^T B \Delta x \\ \Delta x &= AB\Delta x \\ A &= JJ^T \\ B &= A^{-1} \end{aligned}$$

2.4 Dynamics Based Control

This approach computes the set of actuators values and muscle forces needed in order to get an optimal solution to the goal. This process takes place in simulation time, or, in other terms, it's computed *on-line*.

Several examples combine input motion with a dynamic response [9]. In [23], it is used only when an unpredicted physical input influences the character, while the rest of the time it is based on input motion data and balance controllers. [30] uses a flexible muscle-skeletal model that computes the actuation forces from the simulated muscles and optimizes these values to create realistic movements based on the available energy. [31] defines *Spacetime Constraints* and uses these to describe the goal of the character while setting time and space restrictions, as the name suggests, and under some parameters of a function that must be optimized. In another approach, a biomechanics based muscle model is constructed and animated using the dynamic equations of motion [14].

Multibody dynamics describes the relationship between forces acting on a system and the accelerations that they induce. There are two main algorithms that are used to specify the motion of the MB.

Forward Dynamics: It computes the accelerations given the forces applied to the MB. It can be written as a function

$$\ddot{q} = Forward\_Dynamics(q, \dot{q}, \tau)$$

where q is a vector that describes the positions of the joints (angles), $\dot{q}$ is a vector that contains the velocities of the joints, $\ddot{q}$ is the value of the acceleration of the joint and finally, τ is the torque applied to the body.

Inverse Dynamics: Computes the torques needed in order to achieve a desired acceleration. The corresponding function would be

$$\tau = Inverse\_Dynamics(q, \dot{q}, \ddot{q})$$

2.4.1 Equations of motion

The Multibody problem can be solved by different numerical methods. The most popular ones are Euler-Lagrange, Kane's and finally, Euler-Newton. We are going to introduce these various methods in this chapter.

2.4.1.1 Euler-Lagrange

This is a simple approach for non-conservative force systems. The *Lagrangian* is used to find the generalized forces and is defined as follows

$$\mathcal{L} = K - V$$

where K is the total kinetic energy of the system and V is the total potential energy of the system, in terms of joint positions and velocities in generalized coordinates, q and $\dot{q}$ respectively. The *kinetic energy* of the system is found by the formula

$$K = \frac{1}{2}\dot{q}^T H \dot{q} \tag{2.1}$$

And the potential energy is defined by

$$V = -\int_A^B F\, dx = U_B - U_A$$

where A and B are the positions and they define the trajectory of the body. The *Euler-Lagrange* equation of motion is

$$\tau = \frac{d}{dt}\left(\frac{\partial \mathcal{L}}{\partial \dot{q}}\right) - \frac{\partial \mathcal{L}}{\partial q}$$

which gives the generalized joint forces and torques.

The *Lagrangian mechanics* constraint forces are not taken into account, therefore it is more simple and conceptually more easy to understand. An example of dynamic-based animations is [22], in which it is applied for the simulation of flexible characters with a passive dynamic simulation approach. The disadvantage of this method is that the usage of energy functions can easily scale the size of the *Lagrangian* and is consequently inefficient when computing numerical solutions.

2.4.1.2 Kane's method

This method is not as popular as the other two mentioned. One example is [40], in which it is used to produce custom, compact and efficient ODEs. There still are many researches in other fields that do use these equations in robotics [17] and for spacecraft [37][38]. In [35], Kane's equations are reformulated in order to make systematic assembly of large systems of equations much simpler, minimizing its dimensions and being suitable for the computation of numerical solutions. As a matter of fact, because it uses generalized forces, there is no need for analyzing the interactive and constraint forces of the system, which can make *Newton-Euler's* method expensive for large systems; and unlike *Euler-Lagrange's* method, there is no usage of energy functions and thus, no need for differentiation, which can lead to inefficiency [36].

Kane's equations for a system of N bodies with k independent quasi-velocities u_j are

$$F_j + F_j^* = 0 \quad for \quad j : 1, ..., k$$

where F_j is the impressed generalized force and F_j^* is the inertial generalized force. To obtain these values we compute

$$F_j = \sum_{i=1}^{N} [(\frac{\partial v_i}{\partial u_j})^T f_i + (\frac{\partial w_i}{\partial u_j})^T n_i]$$

$$F_j^* = -\sum_{i=1}^{N} [(\frac{\partial v_i}{\partial u_j})^T m_i \dot{v_i} + (\frac{\partial w_i}{\partial u_j})^T (I_i \dot{w_i} + w_i \times I_i w_i)]$$

where v_i is the ith body COM's linear velocity and w_i is its angular velocity, f_i and n_i are the resultants of the impressed forces and momentum, m_i and is the body mass and I_i is represents the moment of inertia.

2.4.1.3 Newton-Euler

Featherstone [15] adapted the equations of motion of *Newton-Euler* in a recursive form that can be applied to Multibody kinematic trees in a very efficient manner.

This approach has been broadly used in many rigid body composed multibody simulations. In [28], an analytical constraint approach is combined with Featherstone's FD recursive algorithm resulting in an efficient method for animating a character. Another example is [33], in which a dynamic filter method is introduced for creating more realistic motions by computing the accelerations of the input motion and removing the forces and torques that would be unfeasible in reality.

The canonical equation of the equation of motion that describes the movement of a body is

$$\tau = M(q)\ddot{q} + C(q, \dot{q}) \tag{2.2}$$

where $M(q)$ is the matrix that describes the position dependent mass distribution, *Inertia Tensor*; $C(q, \dot{q})$ stands for the Coriolis, centrifugal forces, gravity and other forces acting on the system. It is also called the *bias force* and it is the value such of τ such that will produce zero acceleration.

For a Multibody system, the equation of motion takes the form

$$\begin{bmatrix} \tau_1 \\ \tau_2 \\ \vdots \\ \tau_N \end{bmatrix} = \begin{bmatrix} H_1 & 0 & \dots & 0 \\ 0 & H_2 0 & \dots & 0 \\ \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & \dots & H_N \end{bmatrix} \begin{bmatrix} \ddot{q_1} \\ \ddot{q_2} \\ \vdots \\ \ddot{q_N} \end{bmatrix} + \begin{bmatrix} C_1 \\ C_2 \\ \vdots \\ C_N \end{bmatrix}$$

where N is the total number of bodies in the system.

For a constrained system, the equation of motion is modified into

$$H\ddot{q} + C = \tau + \tau_c$$

where τ_c is the *constrained force. Jourdain's principle of virtual power* says that

"The constraint force delivers zero power along every direction of velocity freedom that is compatible with the motion constraints."

Therefore, the implication $\tau_c \cdot \dot{q} = 0$ has to be true.

2.5 Conclusions

The different numerical methods that we introduced in the sections above have different advantages and disadvantages that make them more suitable for some applications than others, depending on the size of the system, the purpose, its simplicity and the energy functions involved. The most popular methods are by far *Newton-Euler* and *Euler-Lagrange*. While the first one is efficient and suitable for real-time simulations, the second one is

preferred for dynamic modelling [38]. The main difference between these methods is how constraint forces and interactive forces are or aren't taken into account in one or another. The former method is very suitable for kinematic trees, as it can be written in the recursive form to avoid re-calculations of some values, thus improving performance. But, if only a few system forces are to be solved it is an inefficient method as it will still have to take all the forces and moment balances of each body in the system. The latter method's main disadvantage is that the *Lagrangian* can be very inefficient to solve for large Multibody systems.

Chapter 3

Relevant Concepts

3.1 Constraints

The *Principle of Virtual Work* states that the total virtual work of external forces F_{ext} that act on the body is zero for any virtual displacement of this. That means that in order to satisfy the constraint of a joint, there is a infinitesimal change in the position which actually does not represent a real displacement. The total force acting on the joint (*net force*) is zero, thus the joint is in equilibrium and therefore the virtual work should also be zero. This means that the constraint forces F_c are actually not contributing to the system but define the motion restrictions that must be satisfied. The constraints, and in particular joint constraints, can be classified as

- *Unilateral Constraints* are referred to as one-sided constraints that are defined to prevent the penetration between two bodies via inequalities.
- *Bilateral Constraints* are two-sided constraints that are defined via equalities.
- *Implicit constraints* are written in the form $C(q) = 0$. These constraints are usually easier to deal with when solving the equations of motion of the system and have to modelled using *Differential Algebraic Equations* (DAE). The DAE solves the implicit joint constraint while satisfying $M\ddot{q} = F_e + J^T F_c$, where J is the Jacobian is the Jacobian of C and F_c are the constraint forces.
- *Explicit Constraints* are written in the form $q = f_c(y)$, where $f_c(y)$ is an explicit motion constraint function of the independent variable y.
- *Holonomic Constraints* constrain the variables of the position of the joint and could be dependent on time.
- *Non-holonomic Constraints* constrain the velocity variables.

There are several methods for solving constrained systems, these are some of them:

- Penalty-based, which uses a spring-damped system.
- Impulse-based, which unifies all types of contact.
- Constraint-based, solving the equations of motion whilst satisfying the constraint definitions.
- Optimization-based using QP optimization solvers (NP-hard problem)

The constrain solvers are invoked at each time step, after applying collision detection and computing the contact regions and forces. Subsequently, they compute the constraint forces that are needed in order to satisfy the constraints, and apply them to the different joints correspondingly before the integrator computes the state of the next frame.

3.2 Joint Types

The different joint types can be defined in a multibody system. The existing 6 degrees of freedom are defined by the rotations and translations along the XYZ axes.

- *Fixed Joint:* A joint that has $\emptyset$ DOFs.
- *Prismatic Joint:* A joint with 1 DOF that provides a linear sliding movement on the specified axis.
- *Revolute Joint:* A hinge joint with 1 DOF that provides rotation in the specified axis.
- *Spherical Joint:* A joint that provides 3 DOFs, one for each axis (X,Y,Z) in the Cartesian coordinate.
- *Planar Joint:* A joint with 3 DOFs that provides a planar movement along a specified rotation axis.

3.3 Physics Engines / Physics Simulators

Physics engines are software that have the purpose of providing an approximate simulation of physical environments. They provide methods for collision detection, numerical integration, soft and rigid dynamics, kinematics computation methods and forward dynamics methods.

In this chapter we will give a small introduction to these concepts and introduce Bullet Physics, the engine that is used for the solution that we are implementing. Specifically, we will be using the last release which is the Bullet Physics SDK 2.83 [18].

3.3.1 Collision Detection

Collision detection gives solution to the problem of detecting the intersection of two or more objects. For each time step in the simulation this algorithm is called to detect the collisions. It provides the information needed in order to compute the *collision response*, which will determine the changes in the motion of the objects involved in the collision of the simulation. This information includes:

- *Time of Impact* This might not be provided, as collision detection is only run each time step and this value is therefore the exact moment in which the detection is already happened during this time step. The time of impact might not be accurate in this case, as time steps are a division of the timeline that the algorithm requires as there is nothing such as real-time in simulation applications. The algorithms and physic engines that do provide an estimation of this value via linear interpolations or by rolling back the simulation.
- *Velocity* It returns the velocity of each object involved in the collision at the moment of the impact. This velocity is calculated with the following formula, which takes into account the current position and the previous position into account:

$$v_k = \frac{q_k + q_{k-1}}{\Delta t} \tag{3.1}$$

  Where q_k is the is the vector containing the angles of the joints of the object at time step k. In a real time simulator this is often translated into a semi-implicit integration step

$$v_{k+1} = v_k + \Delta t a_k \tag{3.2}$$

- *Acceleration*

  The accelerations of the joints of each object are related to the current and previous and next pose accordingly to the following formula

$$a_k = \frac{q_{k+1} - 2q_k + q_{k-1}}{\Delta t^2} \tag{3.3}$$

  This formula is translated in the same fashion as the velocity as

$$q_{k+1} = q_k + \Delta t v_{k+1} \tag{3.4}$$

For efficiency, the algorithms usually use *bounding boxes* instead of the real shapes of the objects. These boxes are normally simple shapes that encapsulate the whole object. The algorithm, therefore, only has to take the boundaries of these boxes into account. In Bullet Physics we apply *btCollisionShape* to the collider objects. In our case, the collision detection is in this regard exact, because we are using the exact same shape for the box as for the collider object.

3.3.2 Numerical Integration

The *Initial Value Problem* searches for a function that satisfies the relation between an unknown function and its derivatives. That is solving the differential equation. The behaviour of the system is described by an *Ordinary Differential Equation ODE* of the form

$$\dot{x} = f(x, t)$$

where x is the *state* of the system and $\dot{x}$ is its time derivative. The problem is described by giving the ODE and an *initial value* of the unknown function at a given time t_0. The solution of this problem specifies the evolution of the system with time.

3.3.3 Rigid Body Dynamics

This dissertation does not concern any soft dynamics and therefore, the latter will not be discussed.

An MB can be interpreted as a system of particles. Each particle is part of the rigid bodies that form the MB. This simile is applicable because a rigid body also has a mass, its state can also be described by its position and its velocity and acceleration at a time t and it can be influenced by external forces. The main difference is that a rigid body does have a *spatial extent*, which will induce to another set of concepts that we will discuss later on in this chapter.

Some notes on notation that we will be using henceforth. We will write the velocity of a rigid body as

$$v(t)$$

Another important concept to take into account when modelling rigid bodies is the *Linear Momentum*, defined as

$$P(t) = \sum_{i=0}^{N-1} m_i \dot{r_i}(t)$$

where $\dot{r_i}$ is the total velocity of the ith particle, which is defined as

$$\dot{r_i} = v(t) + w(t) \times (r_i(t) - x(t))$$

The main difference is that a rigid body does have a *spatial extent*. This introduces a few new concepts that are relevant for the computation of the dynamics of the rigid body.

3.4 Multibody Dynamics

In this dissertation we are going to attempt to implement a fully physical approach. That means we are going to rely on forward and inverse dynamics to control the motion of our character. For this, we are going to need to introduce some concepts. These notes are based on Witkin and Baraff's Physically Based Modelling Course notes [3].

A *Multibody* MB can be seen as a system of bodies attached by *joints*. Each body can, at the same time, be interpreted as a *particle* that has *mass* and *velocity*. The total mass of the system would then be defined by the sum of the masses of the bodies that are part of it.

$$M = \sum_{i=0}^{N-1} m_i$$

The MB has a *Center of Mass* (COM), which in Bullet Physics will be defined as the *base* of the MB. The COM, in *world space*, is defined as

$$COM = \frac{\sum_{i=0}^{N-1} m_i r_i(t)}{M}$$

where m_i is the mass of the ith body of the system and r_i is its position in world space. This value can be computed at time t as

$$r_i(t) = R(t) r_{0i} + x(t)$$

where $R(t)$ is the rotation transformation matrix, r_{0i} is the position of the ith particle at the beginning of the simulation and $x(t)$ is the position of the COM at time t.

The COM is defined to be the origin of the *body space / local space*, which means that the positions of the other bodies in the MB in local space are relative to the COM. These values r'_i (following Witkin and Baraff's notation) can be calculated with either of these following formulas:

$$r'_i = r_i(t) - x(t)$$
$$r'_i = R(t) r_{0i}$$

To represent the *state* of the MB we need to define

- *position vector* q
- *velocity vector* $\dot{q}$
- *acceleration* $\ddot{q}$

For the different degrees of freedom of each joint, q can be an angle (in spherical or revolute joints) or a translational value (in prismatic joints). The velocities and accelerations are their first and second time derivatives, respectively.

Chapter 4

Research Methodology

This chapter will have the following structure: Firstly, a description of the work that is going to be analyzed and discussed in the following chapters. Secondly, a methodology that was proposed before the decision to change the direction of this research and lastly, the reasons for which this decision was taken. The previous methodology is introduced in order to understand the proposals for future work that will be discussed in the conclusions chapter.

4.1 Inverted Pendulum

As the main purpose of the research is to seek for plausible movements, the most basic pose that one has to find is the upright standing and balanced pose. For that, this paper is going to focus on the balance of an inverted pendulum with two joints and a weight at the top. In order to demonstrate the effectiveness of the methodology, the base of the pendulum will have two possible states:

- Fixed: an external force can be applied to the joints and the weight but will not disturb the base.
- Uniform movement in the X axis: a motor will control the movement of the base and apply a uniform velocity to it.

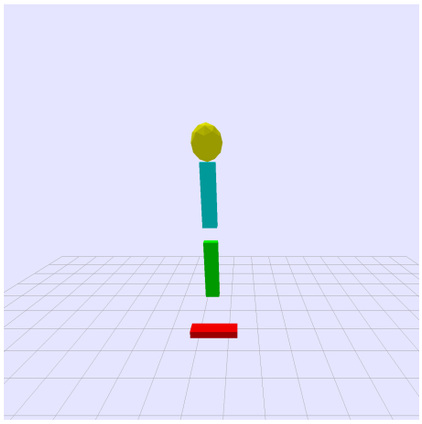


Figure 4.1: Inverted pendulum model

4.1.1 PID controller

In order to reach balance, a PID (proportional integral derivative) controller will be applied at each timestep. The controller will compute the error of the previous timestep and take it into account for the currrent one. Error is defined with the following formula.

$$Error = DesiredPosition - ActualPosition$$

As the name of the controller suggests, P stands for proportional, I stands for integral and D for derivative. The controller has the three corresponding variables that are to be set in order to control the output value. The proportional variable K_d controls the weight of the current error, whilst the integral variable K_i controls the past errors that are accumulated over the past timesteps. Finally, there is K_d that gives weight to future error that is computed as

$$FutureError = Error - PreviousError/(\Delta time)$$

The result that the controller returns is the torque that needs to be applied to the corresponding joint in order to reach the desired position. In order to get a realistic value, the maximum torque that can be applied can be restricted, as well as a minimum torque.

4.2 Original methodology

In order to simulate a physical character we need to construct it first. For this, we will use a *btMultibody* object defined in Bullet Physics. This provides the construction of as many links as we need. We will construct a very simplified version of a human based body.

Similar to reality, in a physics based model, the character is controlled through *forces* and *torques* generated by *actuators*, which are motors that are responsible for controlling motion. This might give an unrealistic motion to the character, as these come from external manipulation of the body, and so it appears as if the body was a marionette. In order to prevent that, we need to add constraints to the forces and torques that are applied to the model and modify its equations of motion.

The purpose of this animation is to make the character appear to have a goal and be able to maintain balance during the journey. As such, our approach will be using inverse dynamics in order to compute the actuator values. These will be optimized by using constraints and objective functions, thus the joint torques will be obtained through *constrained optimization*. We can divide the computation of the dynamics in two steps:

1. Optimization

2. Forward Dynamics

An example of this combination of dynamics and optimization can be found at [47], in which they add a third step between 1 and 2, because of the parameters that are optimized in their cost functions. We optimize the internal torques of the joints while they optimize the joint accelerations.

4.3 Modelling the character

In this chapter we will explain how the body is constructed and linked and how we are going to define the different constraints within the joints. Most of the definition of the model follows Featherstone's scheme in order to make the model compatible with his forward dynamics algorithm for articulated bodies.

4.3.1 Definition of the Model

As we mentioned before, our body is a simplified version of the human body. Our definition will contain only 34 DOFs and 14 joints, whereas the real human body has 244 degrees of freedom and around 230 joints.

In Bullet Physics there is an existing class called *btMultibody* in order to create an MB. The different bodies that conform the MB are called *links* and are objects of the class *btMultibodyLink*. The joints determine the kinematic relationship between each pair of rigid bodies and for this Bullet Physics implements the different joint types defined by Featherstone [15].

In order to apply the inverse kinematics in an efficient way, Bullet Physics recently implemented part of the MB conversion into a *kinematic tree*. The latter meaning that MB's connectivity graph is topologic. Therefore there only exists one path between any two pair of nodes in the graph. We will follow a particular scheme for numbering the joints and bodies of the MB. This scheme is called *regular numbering* [15] and is described as follows:

1. Choose a spanning tree G_t, subgraph of G that contains all of its nodes and any subset of arcs such that G_t is topological.

2. Assign the number $\emptyset$ to the node representing the fixed base and define this node to be the root of G_t.

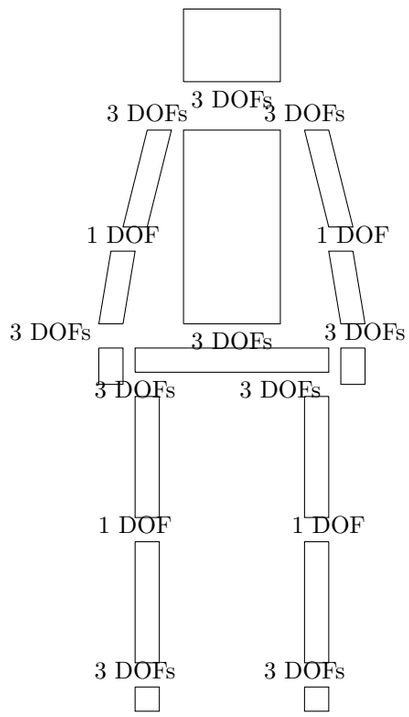
3. Number the remaining bodies from 1 to N_B in order such that each node has a higher number than its parent, where N_B is the total number of bodies of the MB.

4. Number the arcs in G_t from 1 to N_B such that arc i connects between between node i and its parent.

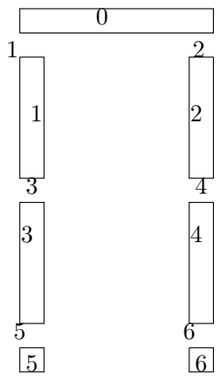
5. Number all the remaining arcs from $N_B + 1$ to N_J in any order, where N_J is the total number of joints of the MB.

6. Each body gets the same number as its node, each joint gets the same number as its arc.

An example of the application of this scheme to a model is shown in the figure below.



(a) Multibody structure. The joints with 3 DOFs are spherical whereas the ones with only 1 DOF are hinges/revolute joints.



(b) Example of a numbering of the joints and bodies of the Multibody following the *regular numbering* scheme.

The Multibody will have the following bodies and joints:

Body Parts

- Head
- Body/Spine
- Left Upper Arm
- Right Upper Arm
- Left Lower Arm
- Right Lower Arm
- Left hand
- Right hand
- Pelvis
- Left Upper Leg
- Right Upper Leg
- Left Lower Leg
- Right Lower Leg
- Left Foot
- Right Foot

Body Joints

- Neck
- Left Shoulder
- Right Shoulder
- Left Elbow
- Right Elbow
- Left Wrist
- Right Wrist
- Coxis
- Left Hip
- Right Hip
- Left Knee
- Right Knee
- Left Ankle
- Right Ankle

We have to be aware of the fact that, as this is a simplified representation of the human body, the movements will also be limited in the sense of realism.

4.4 Constrained Optimization Controller

As mentioned before, there are several methods for solving constraints and several types of constraints that can be more suitable than others depending on the method used for solving the equations of motion of the system.

Although the penalty-based method is the most popular, there are some downsides to it. The advantage of the method is that is computationally cheaper than most of the other approaches. However, this method is not trivial, though simple to understand and implement, and as a matter of fact, there are many kinds of algorithms for computing the contact forces and contact regions exist in order to simplify the task. Nevertheless, a great amount of parameter tuning is involved in the process and it can suffer from secondary oscillation problems.

Moreover, because our approach attempts to let the character reach balance by using optimized torques on the joints that simulate the activation of certain muscles, it is more intuitive to make usage of an optimization controller.

The controller solves the problem of finding the optimal parameters to the objective functions that we will define later on. These objective functions will be subject to the set of constraints defined to restrict the joints from unnatural looking or unfeasible motions. The result should be a set of joint torques that is optimal given the goals and constraints at each timestep.

4.4.0.1 Constraint Definitions

The most important constraint that must be satisfied is the one that simulates the dynamic laws of the motion. That is, the Newton-Euler equation of motion:

$$0 = M(q)\ddot{q} + C(q, \dot{q}) - \tau$$

In order to create a natural looking motion, it is ideal that the joints are constrained by biomechanical constraints.

A great amount of constraints are covered by collision detection and only a flag is required in order to avoid penetration, in most physic engines, such as Bullet Physics. Therefore, we will only focus on the definition of the constraints that concern the motion of the joint. For example, elbows and knees don't bend backwards in real life. As the figure of the model above shows, these two joints consist of only one degree of freedom, a hinge at the z axis. These constraints express the legal positions of the joint and can be defined as:

$$q_{knee} \geq 180^{\text{o}}$$
$$q_{elbow} \leq 180^{\text{o}}$$

To avoid non-natural looking motions of the joints, a set of constraints in terms of their velocities must be defined.

$$-\dot{q}_{i\,max} \leq x_i \leq \dot{q}_{i\,max}$$

The internal forces of the system may imitate a musculoskeletal model's behaviour. That is, there will be a maximum internal torque that each degree of freedom of each joint can reach. The total internal torque of the joints will be optimized in order to define a unique solution that gives values to these parameters.

In order to avoid null values for the virtual "muscle" forces when optimizing the system, the set of moment equations must be part of the constraint system. These are

$$M_i = d_{ij} \times \tau_{im}$$

where $\tau_i m$ is the force of the mth muscle of the ith body and d_{ij} is its moment arm with respect to the jth joint. Moreover

$$\tau_i \geq 0$$

$$\tau_i \leq PCSA_i \omega_{max}$$

where ω_{max} is the maximum muscle stress and $PCSA_i$ is the physiological cross-sectional area of the muscle.

For the different actions, there are different sets of constraints that have to be taken into account. Our interest is to let the character maintain balance. For this, the *Zero Moment Point* ZMP, which specifies the point so that the reaction force at the position of the foot that makes contact with the floor produces zero moment, must be satisfied. We can define an inequality to ensure that the projection of the COM does not fall too far from the ZMP. The ZMP is the height of the floor. The rest of the coordinates are defined as [43]:

$$ZMP_x = \frac{MgP_x + ZMP_z\dot{P}_x - \dot{H}_y}{Mg + \dot{P}_z}$$
$$ZMP_y = \frac{MgP_y + ZMP_z\dot{P}_y - \dot{H}_x}{Mg + \dot{P}_z}$$

where M is the total mass of the multibody, g is gravity, P is the linear momentum about the COM, H is the angular momentum about the COM and $\dot{P}, \dot{H}$ are their respective derivatives.

In order to maintain balance, the ZMP should lie within the support polygon SP, which is defined by the contact point positions of the feet. The projection of the COM can fall out of this polygon, but if it is too far, there is a great probability that the character will fall. The ZMP is known, and the projection of the COM COM_{proj} can also be computed at each time step. We can define a vector d_{max} that contains the maximum distances permitted.

$$|ZMP - COM_{proj}| \leq d_{max}$$

4.4.1 Optimization and Objective Functions

In order to simulate a musculoskeletal behaviour, there are several cost functions that can be defined and optimized during the integration [44].

- Minimize total muscle stress of the multibody.

$$Z = \sum_i (\frac{F_i}{PCSA_i})^2$$

  where $PCSA_i$ is the cross-sectional area of the muscle.

- Minimize the work done
- Minimize the fatigue damage
- Minimize mechanico-chemical energy

These functions can be optimized in a non-linear form so that it simulates a synergistic activity and so, instead of stressing a lower number of muscles, it shares the stress on a larger selection of them.

Another reinforcement for balance maintenance is to minimize the value of the angle between the axis of the moment of inertia of the multibody and the normal of the floor. That is applicable only if the floor does not have too many small irregularities.

Because our approach is goal-driven and we are employing inverse dynamics, we need to specify the objective function.

4.5 Findings and problems

In this section we will discuss the main problems that were encountered when attempting to apply the discussed methodology will be discussed, as well as some problems with the understanding of the usage of the physics engine library and other issues.

In order to apply the torques and simulate a circular gait, in Featherstone's fashion, we tried to transform the body shown in the figure of section 4 into a kinematic tree. The method in Bullet Physics failed to do so and after debugging and doing some research, it seemed that the implementation was too new and there were some parts of the code which were not yet implemented or were partially implemented. In particular, only prismatic joints where implemented. Because only one axis of motion was specified in the function, instead of using rotating joints, we tried using hinges in order to do a workaround, in order to avoid having to add a matrix that would imply the change of many other functions.

Other findings in these functions where the naming conventions. The kinematic tree constructor calls a function named *addBody*, which at the same time calls a function in the *MultibodyTree Init Cache* with the same name. This function also lacks the definition of spherical and hinge joints and their respective degrees of freedom. Here, in order to simulate the spherical joint with 3 hinge joints we would have to define their parents to the previous links number, so they would all have the same. The implementation was unfinished as the library was hard to understand in general, and there was none or poor documentation on the majority of it. Moreover, the inverse dynamics library was just updated recently, and little information about it is posted on the Internet. The library was incomplete and there were inconsistencies that were used as patches to cover some of the unimplemented methods.

The originally proposed methodology is a result of a long research that started with the interest in inverse kinematics. As mentioned in the introduction, there are several fields of which one has to have knowledge about in order to be able to understand and implement realistic character animation. Specially when it is physics based, because then one requires a knowledge base in biomechanics, differential mathematics, etc. Being the problem of character animation still open and having such a wide range of solutions that differ so much from one another can lead to a lot of confusion for a person who is new to the field. The choice of Bullet Physics was based on the programming language rather than the completeness of the library or ease of usage and understanding. The library is in fact very powerful, but the lack of documentation can become a grater issue for someone with no background on the field. Specially the naming conventions can be hard to understand.

Chapter 5

Findings and Analysis

In this chapter, the results of the PID controller for the inverted pendulum to reach balance will be analyzed. Different weights to the corresponding variables will be applied, as well as different maximum torques, in order to find an effective combination of these.

Before jumping into the results, it is important to comment that all of the tables and graphs are for one full cycle of the movement since the external force is applied. The external force is applied manually in the simulation and has different values for each table. What is important to focus on is the velocity of adaptation pendulum with the torques that the controller provides. Another important fact is that there is a damping attached to the movement and the pendulum does not reach balance immediately. If other external forces are being applied continuously, the body will not reach balance. This is due to the computation of movement of the class *btMultibody*. A part from the torques that are applied to the joints, the different forces that come from the physical environment in which the pendulum is found are add up afterwards and result in a damping movement.

Notice that there can be some disturbances and this is due to human error when exerting an external force to the pendulum. These values will be considered outliers and will be eluded in the analysis of the graphs.

5.1 Fixed base parameter analysis

The base of the inverted pendulum will be fixed and the external torques or other forces do not apply to it.

5.1.1 Proportional parameter

Applying different values to the proportional parameter and setting $Kd = 500$ and $Ki = 500$. It can be observed that as the value of kp increases, the error is smaller: The maximum error value is lower as well as the average error value. The fact that the original disturbance applied to the pendulum is exerted manually can explain the non-linear decrease in the error.

The cycle velocity is directly influenced by kp and this is sensical as kp weights the importance of the current state, which can be a new state that has never been taken into account and therefore neither the future or past errors can accomplish the needed value, and this error is carried throughout the whole cycle.

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.039802585063643	0.00635373245027847	4.03188143993637	0.754617845027848
Min	-0.735373	-0.358878	-74.3345	-43.9788
Max	1.17954	0.533876	119.232	55.1182

Table 5.1: Fix base kp = 50

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0299107076831964	-0.000287355502932379	11.8248885003978	3.94891584202068
Min	-0.0236773	-0.0271863	-104.655	-289.737
Max	0.0928539	0.0265169	500	395.278

Table 5.2: Fix base kp = 100

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0178380826077963	0.00388035446933175	9.8223671805887	3.21554183508354
Min	-0.0249577	-0.054245	-147.02	-500
Max	0.0863816	0.0648927	500	500

Table 5.3: Fix base kp = 150

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.0163397301672236	-0.000213275467926014	10.0847358219571	2.79589822227526
Min	-0.0286917	-0.0264303	-173.684	-362.091
Max	0.0949022	0.0241478	439.881	422.043

Table 5.4: Fix base kp = 200

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.0353672649008751	0.00022549676531424	13.2605262287987	3.21935680986476
Min	-0.0276832	-0.0410416	-229.78	-500
Max	0.0749083	0.059205	297.282	277.228

Table 5.5: Fix base kp = 250

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.0186337399817025	0.000809544292171837	15.606936043755	5.01254707716786
Min	-0.0310787	-0.0347517	-283.635	-269.335
Max	0.0900261	0.0374319	500	462.633

Table 5.6: Fix base kp = 300

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.0207220279793158	-0.0013870524286078	12.7952262593477	2.28629449614956
Min	-0.0178443	-0.0594336	-77.0455	-500
Max	0.0637876	0.0392114	500	500

Table 5.7: Fix base kp = 350

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.0176600187754973	0.00241143104580747	10.6096385449483	2.80950151670644
Min	-0.0254749	-0.0604631	-392.273	-500
Max	0.0513998	0.066611	460.195	500

Table 5.8: Fix base kp = 400

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.0213817417350501	-0.000276148935640414	16.8886206007955	5.07632888830549
Min	-0.026052	-0.0366066	-494.928	-393.292
Max	0.0932922	0.0265258	500	500

Table 5.9: Fix base kp = 450

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.00839095674470963	0.00449629820400953	3.74609121447891	1.98033153174224
Min	-0.215392	-0.112797	-119.978	-92.7701
Max	0.290862	0.140877	146.446	71.1635

Table 5.10: Fix base kp = 500

5.1.2 Integral parameter

As mentioned before, this parameter gives weight to the past error and so, contributes to reduce any accumulated error from the desired position after having applied the torque in order to reach the previous desired position, which after the timestep, the inverted pendulum can have reached or not. Thus, it helps reducing the damping effect that can sometimes be observed. The values of the errors do not decrease but are more normalized as the value of Ki increases. This means, the difference between the greates value of error and the smalles value is inversely proportional to Ki.

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0183616544375497	0.00390479711640413	12.1334507995227	3.1535607239459
Min	-0.0236773	-0.0353901	-315.418	-412.374
Max	0.076094	0.0842281	446.322	500

Table 5.11: Fix base Ki = 100

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0211716468165473	0.00519714166937152	10.4317954887828	2.14619374884646
Min	-0.138462	-0.143549	-70.0173	-120.279
Max	0.17814	0.0856158	90.0936	69.9537

Table 5.12: Fix base Ki = 150

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0168476634757358	0.00388426955051709	8.35706658941925	1.46947521559268
Min	-0.112138	-0.129958	-56.8612	-114.646
Max	0.1433	0.0685878	74.567	70.0793

Table 5.13: Fix base Ki = 200

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0109023911105807	0.000178873104295942	5.33539392680988	-0.0933560859188557
Min	-0.069405	-0.177849	-35.2551	-160.513
Max	0.103665	0.0617426	82.873	55.7531

Table 5.14: Fix base Ki = 250

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0124314919244233	0.00178118368098648	6.14818291821795	0.371656764677803
Min	-0.100465	-0.135336	-51.1674	-117.453
Max	0.126836	0.0630905	72.9829	58.5967

Table 5.15: Fix base Ki = 300

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.020835845618934	0.00741079152585522	10.5350736976929	3.56852358154337
Min	-0.126201	-0.0737884	-64.5516	-75.3773
f **Max**	0.161594	0.0763203	82.5806	40.6317

Table 5.16: Fix base Ki = 350

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0230576417597454	0.00120977074463007	11.4765134128878	0.0478883078758921
Min	-0.180055	-0.240434	-92.7059	-172.036
Max	0.233682	0.108268	120.195	130.198

Table 5.17: Fix base Ki = 400

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0127761905122992	0.00205369479949085	6.39854095999998	0.602448873731102
Min	-0.101569	-0.123505	-52.1571	-117.552
Max	0.128903	0.060044	75.3395	83.2458

Table 5.18: Fix base Ki = 450

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0115002591567223	0.00193827365950677	5.70608957358791	0.590719452665074
Min	-0.102239	-0.114402	-52.5035	-104.996
Max	0.129425	0.060805	69.2654	61.9304

Table 5.19: Fix base Ki = 500

5.1.3 Derivative parameter

This parameter intends to predict the error value in the next timestep and reduce it. As it can be observed, the difference between the maximum error value and the minimum error value is decreased as Kd increases. The reason is its own definition: it prevents the current torque to generate a greater error in the next timestep.

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0294416471288783	0.00928618767462211	15.203345424821	3.82214272474145
Min	-0.0913749	-0.0895817	-48.8584	-193.318
Max	0.156505	0.0727686	116.909	107.977

Table 5.20: Fix base Kd = 100

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0238739395290374	0.00966755000795544	12.7668694232299	4.15687089976134
Min	-0.05699	-0.0412482	-33.8322	-183.845
Max	0.111317	0.0515724	123.292	74.8627

Table 5.21: Fix base Kd = 200

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0304339887820207	0.0103362082641209	16.0558648377088	5.92946872848052
Min	-0.0637274	-0.0275018	-47.5593	-155.403
Max	0.143229	0.0571142	110.205	105.139

Table 5.22: Fix base Kd = 250

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.036832999443914	0.0103523591786714	19.998554061257	10.1596238961018
Min	-0.067561	-0.0210084	-58.2447	-46.3987
Max	0.169682	0.053724	147.755	244.562

Table 5.23: Fix base Kd = 300

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0331299291567224	0.00511019974470963	18.1619683929196	6.00002331845664
Min	-0.048614	-0.0243757	-51.1552	-263.873
Max	0.137919	0.0337816	163.933	146.629

Table 5.24: Fix base Kd = 350

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0265967180389817	0.000685769966141606	14.9055665485283	4.4751328547335
Min	-0.0344481	-0.0219765	-54.1433	-240.685
Max	0.11846	0.0129678	259.887	185.413

Table 5.25: Fix base Kd = 400

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0211275614272076	0.0000214437059029435	15.1328723070485	4.56442954972156
Min	-0.0230004	-0.0189452	-48.6926	-321.786
Max	0.0832302	0.0186572	312.065	398.365

Table 5.26: Fix base Kd = 450

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0485509198727128	-0.0000440131590771679	34.7777649186953	17.7234042297534
Min	-0.0766239	-0.0261962	-213.222	-218.235
Max	0.199294	0.0448351	500	500

Table 5.27: Fix base Kd = 500

5.2 Uniform movement base parameter analysis

The base of the inverted pendulum will have a motor that applies a uniform acceleration to it, reaching a desired velocity and producing an external force that influences the movement of the pendulum itself. The pendulum suffers from large disturbances when changing the direction of the velocity, which happens in the middle of the cycle.

5.2.1 Proportional parameter

Applying different values to the proportional parameter and setting $Kd = 500$ and $Ki = 500$. In these simulations there is no manual disturbance. The disturbances are generated by the acceleration that the body suffers from the motor attached to its base. Even so, we can observe that the average value of the error is not decreasing linearly. This may be by the fact that in these simulations, the same maximum force value was applied and as can be seen in the tables and the graphs, the needed torque was had a greater absolute value than 500, and so, the errors were greater at those points. This means that balance is also reached more slowly, as there is still disturbance from the acceleration and smaller torques than needed are applied, thus generating a carried error to the next timesteps.

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.06771588157677	0.00249118161667462	2.00385332728718	1.05476571758155
Min	-0.194693	-0.0779817	-500	-500
Max	0.479576	0.0971204	500	500

Table 5.28: Uniform Movement base kp = 100

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.00445812568019089	0.00203087576836118	-2.16481668070804	-0.601190194749401
Min	-0.275844	-0.0689828	-500	-500
Max	0.383916	0.102315	500	500

Table 5.29: Uniform Movement base kp = 150

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	-0.00300519298408906	-0.00126900626848051	-3.90203745902944	-1.76809280461416
Min	-0.190237	-0.0553848	-500	-500
Max	0.408614	0.0941796	500	500

Table 5.30: Uniform Movement base kp = 200

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.00858887499427207	0.00203943719726333	-4.14279004852824	-1.71786701264916
Min	-0.1902	-0.065282	-500	-500
Max	0.393789	0.0900583	500	500

Table 5.31: Uniform Movement base kp = 250

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	-0.00833056832299127	0.001841085648035	-7.02230052601431	-2.51357063261735
Min	-0.217309	-0.0690982	-500	-500
Max	0.352056	0.101286	500	500

Table 5.32: Uniform Movement base kp = 300

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.00396415735894988	-0.000518809190843277	-5.22726928440733	-1.98691552187748
Min	-0.166074	-0.0828832	-500	-500
Max	0.380032	0.0911456	500	500

Table 5.33: Uniform Movement base kp = 350

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	-0.00496899997692917	-0.0015343429664121	-5.21731537740653	-2.12046078265712
Min	-0.186516	-0.0813411	-500	-500
Max	0.355075	0.0830831	500	500

Table 5.34: Uniform Movement base kp = 400

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.00979414142927602	-0.00167190709207637	-3.81435728846459	-1.66368864343675
Min	-0.174145	-0.10867	-500	-500
Max	0.379737	0.0760599	500	500

Table 5.35: Uniform Movement base kp = 450

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.00376726967645187	0.0000711453758949889	-4.17732244391408	-1.56333411217183
Min	-0.17396	-0.0756231	-500	-500
Max	0.347811	0.0633273	500	500

Table 5.36: Uniform Movement base kp = 500

5.2.2 Integral parameter

The integral parameter was tested to have different values while $kp = 500$ and $Kd = 500$. The same problem with the maximum force can be observed in both tables and graphs. What can be observed is that errors seem to be directly proportional to Ki's value: the smaller the value of the parameter, the smaller the value of the errors, in average. But this could be not reliable, as the problem with the torque value influences the results.

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	-0.000926021805568828	0.000637103664105012	-6.09494764614162	-2.28085198011138
Min	-0.165578	-0.0639254	-500	-500
Max	0.313375	0.0708084	500	500

Table 5.37: Uniform Movement base Ki = 100

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.000407301254574385	-0.00214539163100239	-5.26550974940334	-2.5632476246619
Min	-0.153861	-0.0972358	-500	-500
Max	0.350764	0.0837759	500	500

Table 5.38: Uniform Movement base Ki = 150

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.000390199997120134	0.000509512686308671	-5.45940449132856	-1.91604942816229
Min	-0.172632	-0.078184	-500	-500
Max	0.341872	0.0664338	500	500

Table 5.39: Uniform Movement base Ki = 200

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.00223663675226729	0.00120525323684964	-6.31617619474941	-2.61469357915672
Min	-0.163372	-0.0620088	-500	-500
Max	0.342188	0.0774127	500	500

Table 5.40: Uniform Movement base Ki = 250

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	-0.00171970819013525	-0.0000248576745743825	-6.31007546396182	-2.41345848178202
Min	-0.169552	-0.0831381	-500	-500
Max	0.363722	0.0611231	500	500

Table 5.41: Uniform Movement base Ki = 300

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	-0.00526595710342085	0.000687538982012731	-7.21241736276849	-2.86712691544948
Min	-0.160457	-0.0755485	-500	-500
Max	0.326968	0.0781366	500	500

Table 5.42: Uniform Movement base Ki = 350

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	-0.00251006922171837	0.00105879076243437	-6.03201537915672	-2.51521263101034
Min	-0.178348	-0.0657775	-500	-500
Max	0.389339	0.0891645	500	500

Table 5.43: Uniform Movement base Ki = 400

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.00123048122195703	-0.000752824376690534	-5.19681869336515	-1.86808238822593
Min	-0.186929	-0.0826097	-500	-500
Max	0.326192	0.0690439	500	500

Table 5.44: Uniform Movement base Ki = 450

	Error joint 1	**Error joint 2**	**Torque joint 1**	**Torque joint 2**
Avg	0.0115002591567223	0.00193827365950677	5.70608957358791	0.590719452665074
Min	-0.102239	-0.114402	-52.5035	-104.996
Max	0.129425	0.060805	69.2654	61.9304

Table 5.45: Uniform Movement base Ki = 500

5.2.3 Derivative parameter

The difference between the maximum and minimum error values is inversely proportional to Kd's value. The maximum torque value is still being a problem in this cases.

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	-0.0145291231567224	-0.00562425664120922	-7.38198907875896	-3.20306650119333
Min	-0.238339	-0.109755	-127.307	-302.843
Max	0.490461	0.225205	255.535	248.198

Table 5.46: Uniform Movement base Kd = 100

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	-0.0141190333463803	-0.00542566596738265	-6.84909130867144	-2.95734165107398
Min	-0.227371	-0.103429	-121.273	-335.889
Max	0.449045	0.206388	239.887	348.802

Table 5.47: Uniform Movement base Kd = 150

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	-0.0111678933002387	-0.00383861944391409	-6.41768951917264	-2.72740197517899
Min	-0.207959	-0.0952194	-171.341	-365.911
Max	0.420559	0.191379	252.365	393.497

Table 5.48: Uniform Movement base Kd = 200

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	-0.0138002016181384	-0.0051231083699284	-6.3811089061257	-2.73915220055687
Min	-0.21152	-0.096022	-185.536	-500
Max	0.383133	0.174743	262.421	400.925

Table 5.49: Uniform Movement base Kd = 250

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	-0.00981520862529832	0.000122460807637227	-5.81011617684965	-2.3026131057677
Min	-0.197925	-0.0799563	-295.834	-500
Max	0.394237	0.184958	240.299	406.246

Table 5.50: Uniform Movement base Kd = 300

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	-0.0110845427255 37	-0.00594956900851232	-6.46444013524264	-2.73850491535402
Min	-0.193024	-0.0766889	-417.959	-500
Max	0.360416	0.0724013	399.121	500

Table 5.51: Uniform Movement base Kd = 350

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	-0.0121999201607001	-0.00803745351630868	-5.60815879506762	-2.3067482645187
Min	-0.188115	-0.076502	-310.287	-500
Max	0.367653	0.0723982	500	500

Table 5.52: Uniform Movement base Kd = 400

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	-0.000218877477883858	-0.0007534814674463	-5.01686381543357	-1.73801533842482
Min	-0.192178	-0.0793855	-500	-500
Max	0.331158	0.0700194	500	500

Table 5.53: Uniform Movement base Kd = 450

	Error joint 1	Error joint 2	Torque joint 1	Torque joint 2
Avg	0.0485509198727128	-0.0000440131590771679	34.7777649186953	17.7234042297534
Min	-0.0766239	-0.0261962	-213.222	-218.235
Max	0.199294	0.0448351	500	500

Table 5.54: Uniform Movement base Kd = 500

5.3 Conclusion

It can be observed that when the three values are the same, the pendulum reaches more stability and in less time with both static base and with a uniform velocity. As can be seen in the graphs in the annex, there are some outliers in the torque graphs of the uniform velocity base inverted pendulum. These are the torques that need to be applied when there are large disturbances, so that equilibrium can be reached in the next timestep and thus reducing the damping effect that is produced by the large disturbances cause by the change of velocity direction or an increase in acceleration.

Another very important fact to see is that when either Ki or Kp have smaller values than Kd, the torques that are computed make the pendulum more unstable. This can be explained by the fact that, as the weight on them is smaller, there is a carry on the previous errors to the present one. And in order to reduce the damping, the values of Ki are very important, therefore another reason for the instability.

Chapter 6

Conclusions

6.1 Discussion on the implementation

Bullet Physics, although it is a very powerful library and there is a great amount of classes which permit to create a vast range of simulations for physics, suffers from the lack of documentation.

Alternative tools were found, such as HOTINT or EasyDyn. The former is an environtment that offers a graphical interface for the user (using OpenGL) , a dynamics solver (time integrator), a static solver and a multibody kernel. The solvers are efficient and are based on the reordering of the equations of motion for kinematic trees. The main downsides were that it was an environment for Windows and moreover, although tutorials and workshops are offered, little information about its usage is found. Nevertheless, unlike Bullet Physics, it does not have a forum of user programmers who can suggest new ideas of implementation or solve doubts. On the other hand, EasyDyn is Unix friendly. Although there is a user guide, it still has the same problem, as there are no class reference pages nor a forum to discuss some possible issues.

6.2 Conclusions

The main results of the applied methodology prove that the PID controller's parameters can successfully be tuned in order to reach the balance on an inverted pendulum, with both movement or static. It is also seen that in the cases where it has been applied, the current error and the previous error are most important out of the three values that are taken into account in the computation of the torques for each timestep.

It has to be tuned differently for other different cases. As described before, the parameters do not influence exactly the same in the two cases that were applied to in this paper. Sometimes one parameter would cause greater changes in the error values than in the other situation and viceversa.

The previous proposed methodology of the dissertation remains unfinished due to the reasons stated in the discussion afterwards. In a nutshell, it is difficult to choose a methodology that can apply to the initial idea of the project, as there is a wide range of methodologies that can be followed. As stated previously, it can be confusing for one who is new to the field. It has to be taken into account the fact that the problem still remains open.

6.3 Recommendations and Future Work

An actual implementation to test the different constraints and evaluate the realism of the results could be done by implementing an extension of Bullet Physics libraries, consistently with its notation and existing functionalities. Or modify the existing implementation and write it in a more expressive form, so that the user programmer can find it much more readable. Most importantly is to understand the behaviour of the different components and physical attributes of the classes and provide documentation about those within the programming reference guide.

Furthermore, an analysis of all the existing work could be done in order to compare and contrast the different methods and classify their usages depending on the purpose of the users.

Related to our work, some more in-depth research on biomechanical constraints of muscle usage and optimization functions would be relevant in order to find the *Muscle Activation Patterns* (MAP) that make the motion appear realistic to the human-eye. These constraints can be applied to a model similar to the inverted pendulum, to simulate a pair of human legs maintaining balance while being influenced by external disturbances in a realistic manner, unlike here where the there was a maximum torque applied equally to both joints.

Chapter 7

Annex

Graphs of the different values for each parameter that has been discussed in the chapter on Findings and Analysis. The graphs are going to be sorted by ascending value of the parameters and will be displayed in the following order: first showing the position errors and secondly, the computed torques. The position errors are in *quaternions* and torques are in *Newtons*.

7.0.1 Proportional variable

7.0.1.1 Joint 1

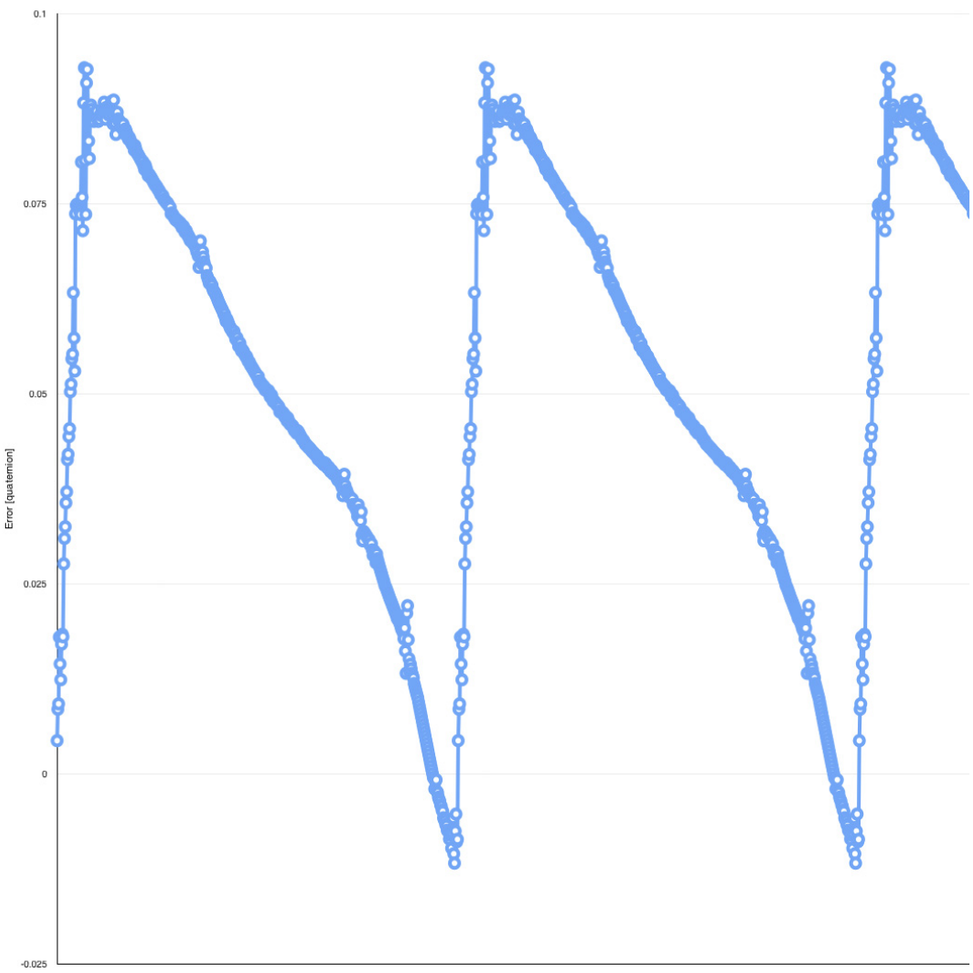


Figure 7.1: Fixed Base Position error, kp = 100

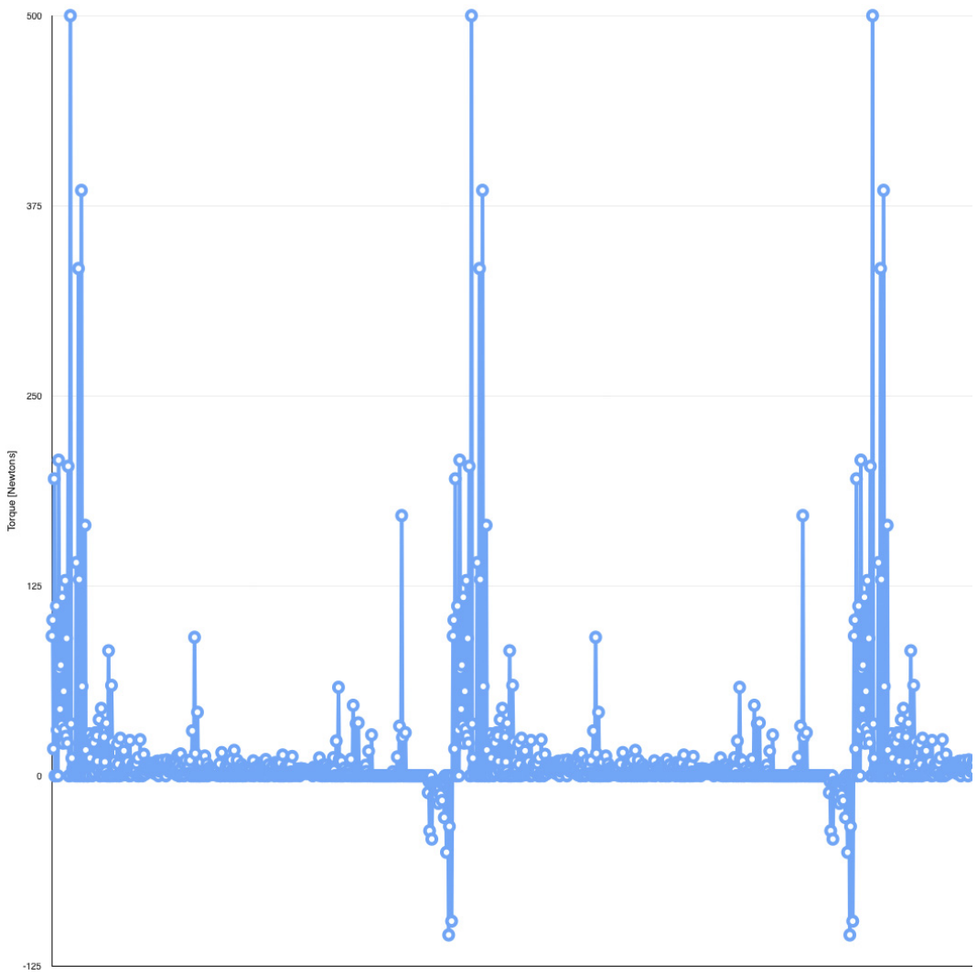


Figure 7.2: Fixed Base Torque, kp = 100

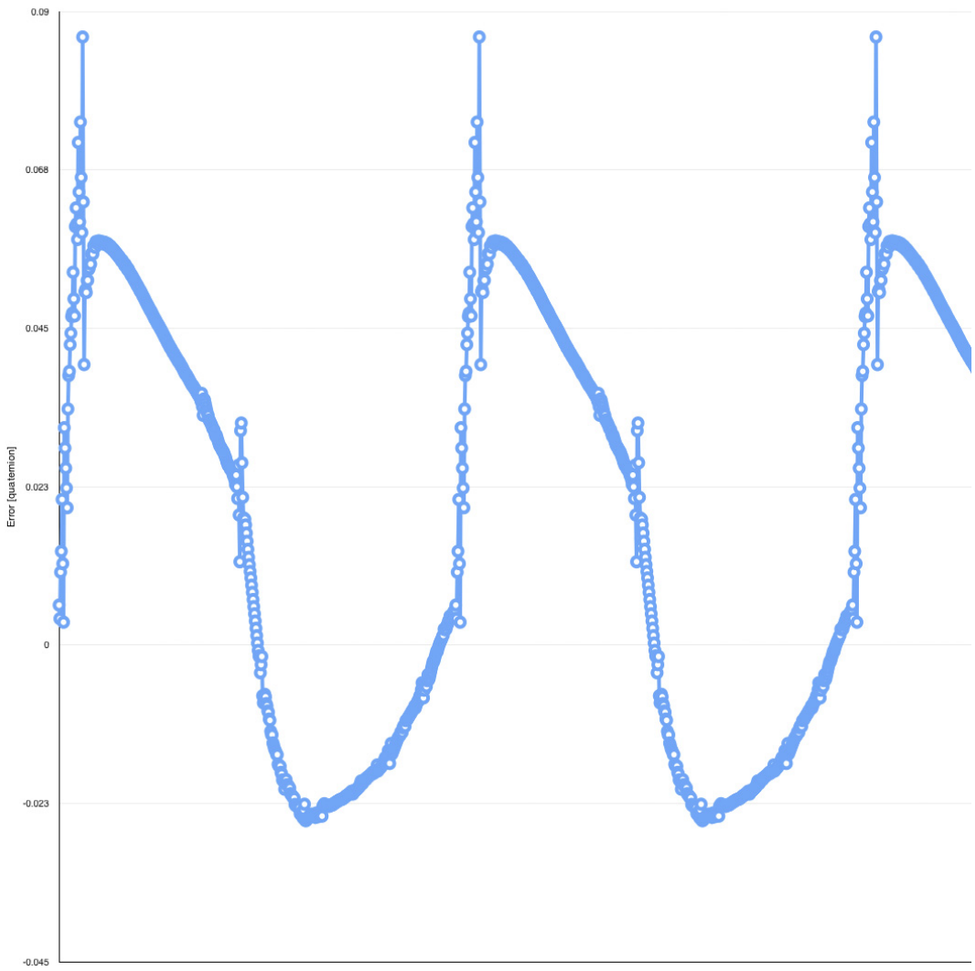


Figure 7.3: Fixed Base Position error, kp = 150

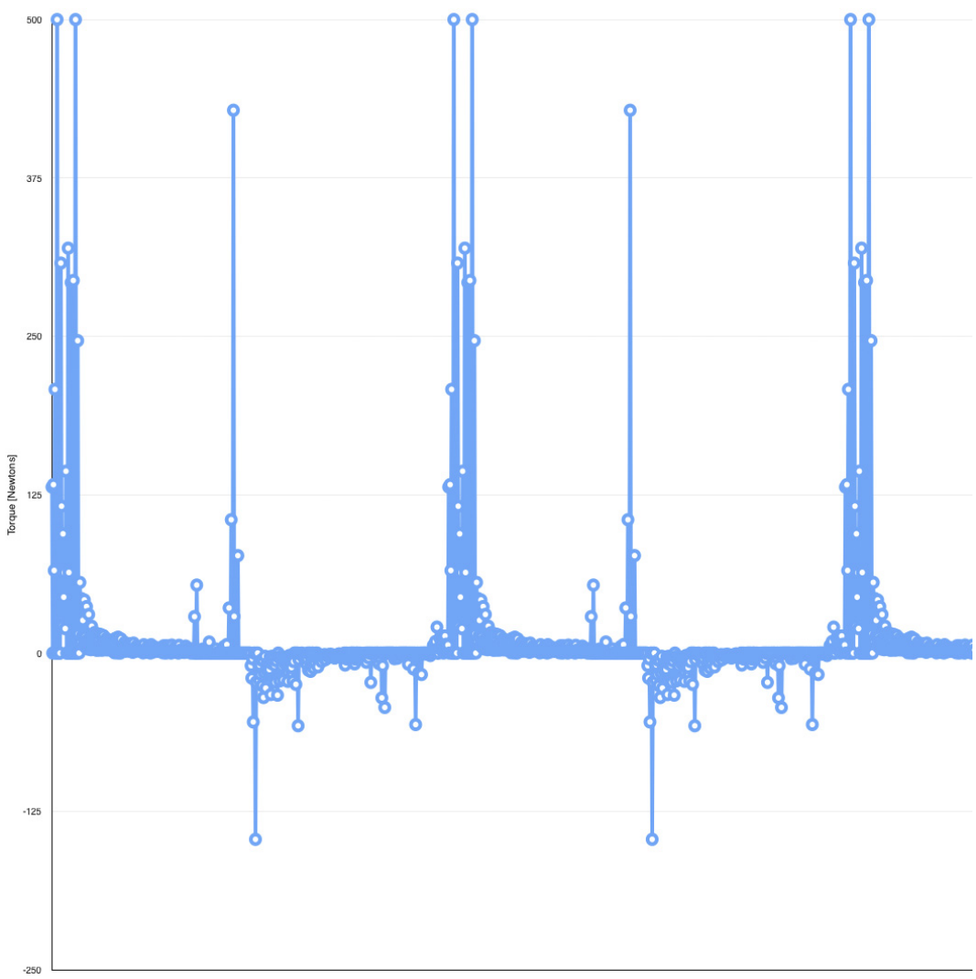


Figure 7.4: Fixed Base Torque, kp = 150

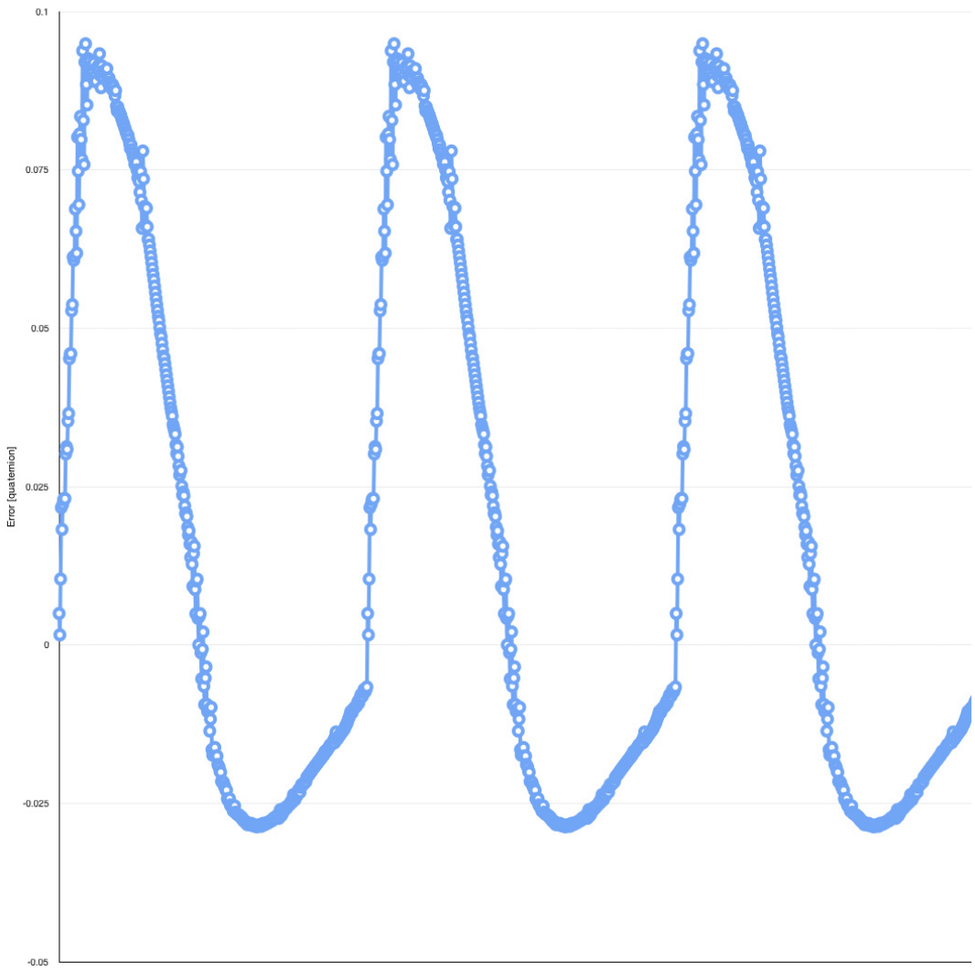


Figure 7.5: Fixed Base Position error, kp = 200

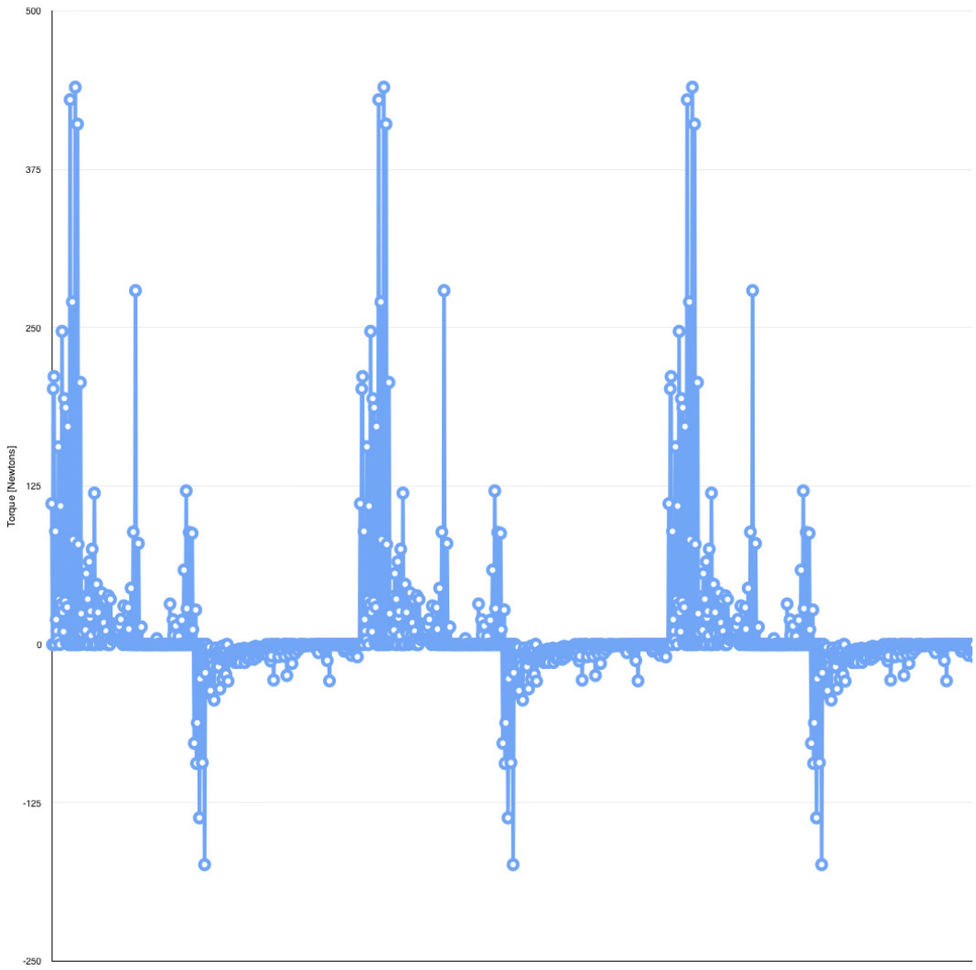


Figure 7.6: Fixed Base Torque, kp = 200

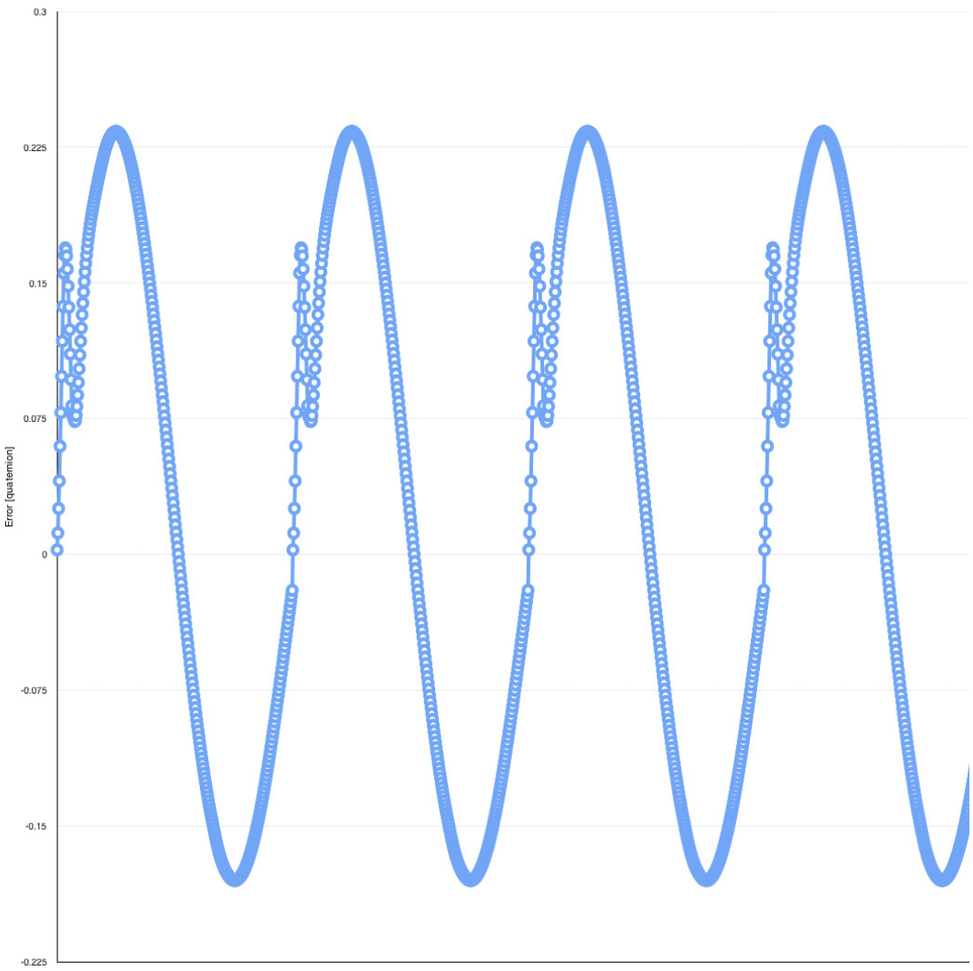


Figure 7.7: Fixed Base Position error, kp = 250

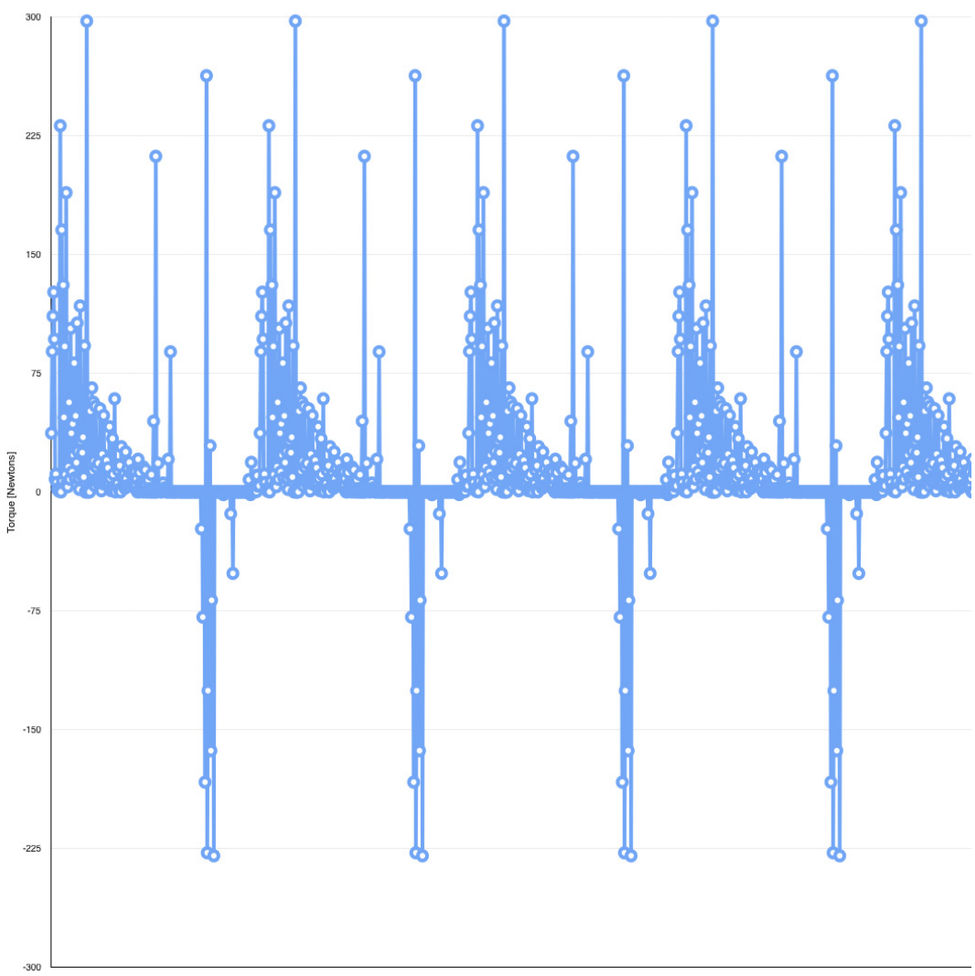


Figure 7.8: Fixed Base Torque, kp = 250

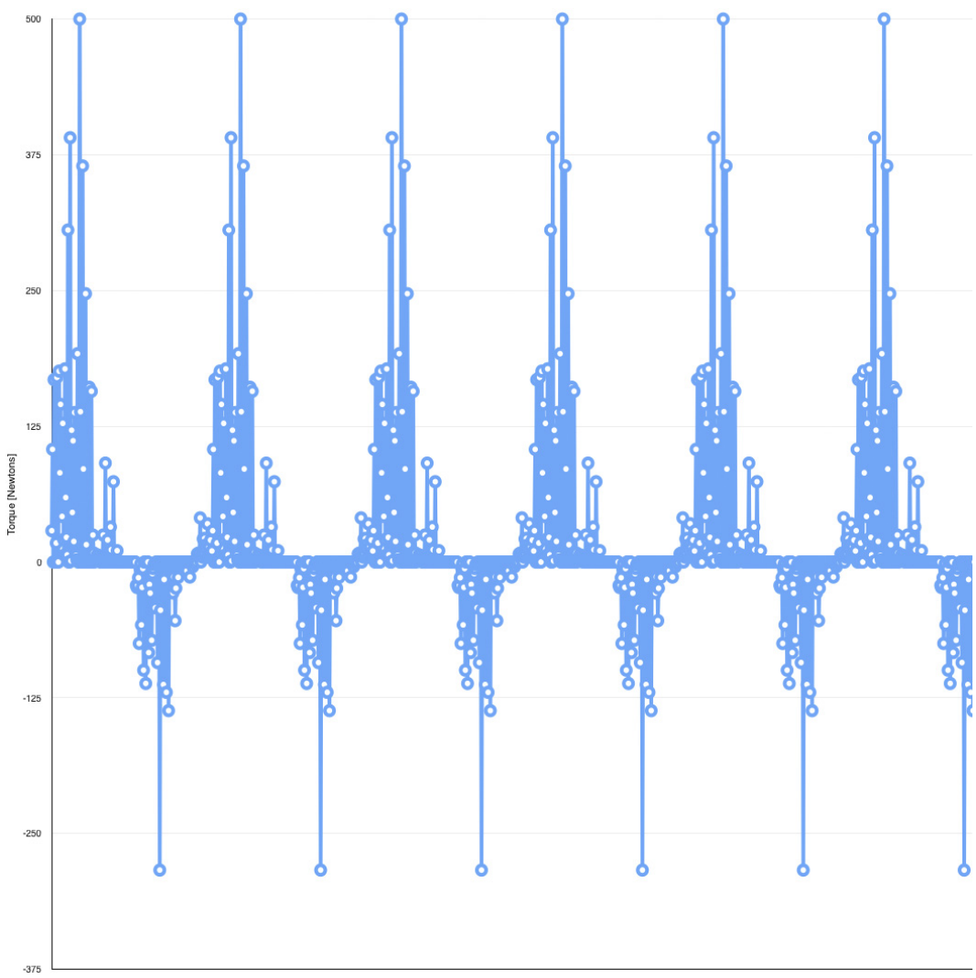


Figure 7.9: Fixed Base Position error, kp = 300

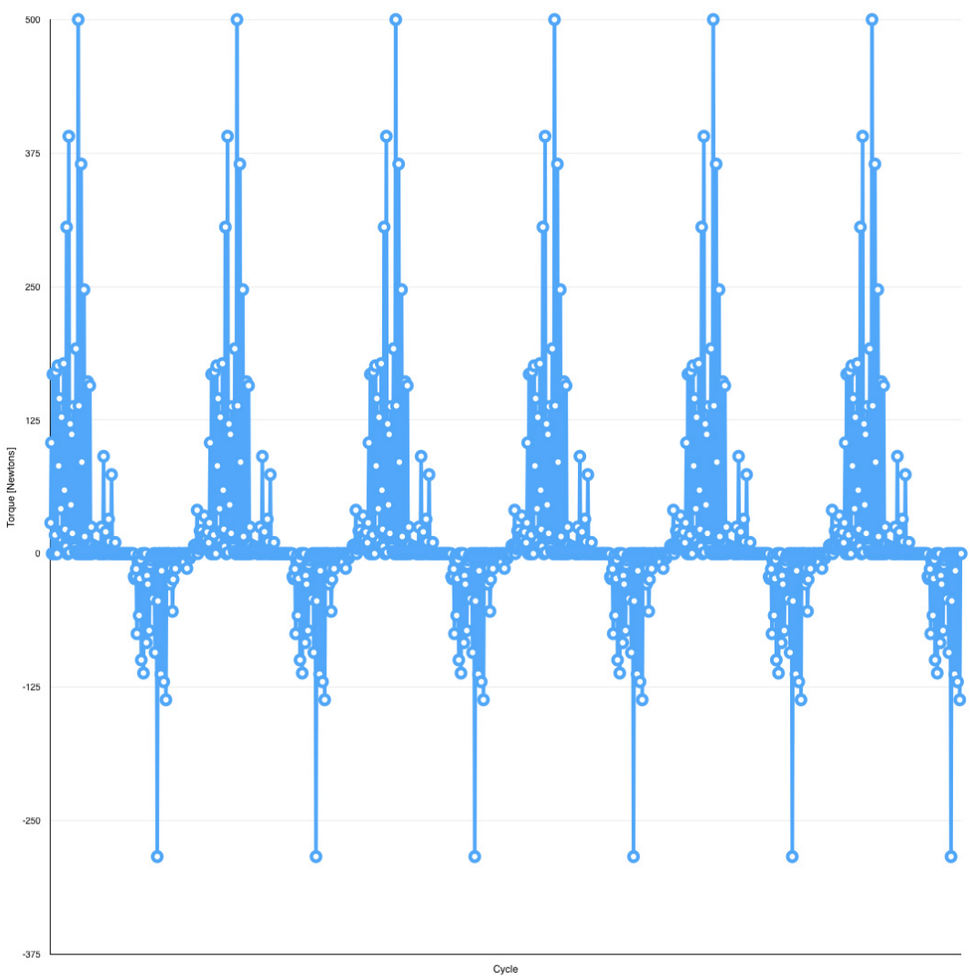


Figure 7.10: Fixed Base Torque, kp = 300

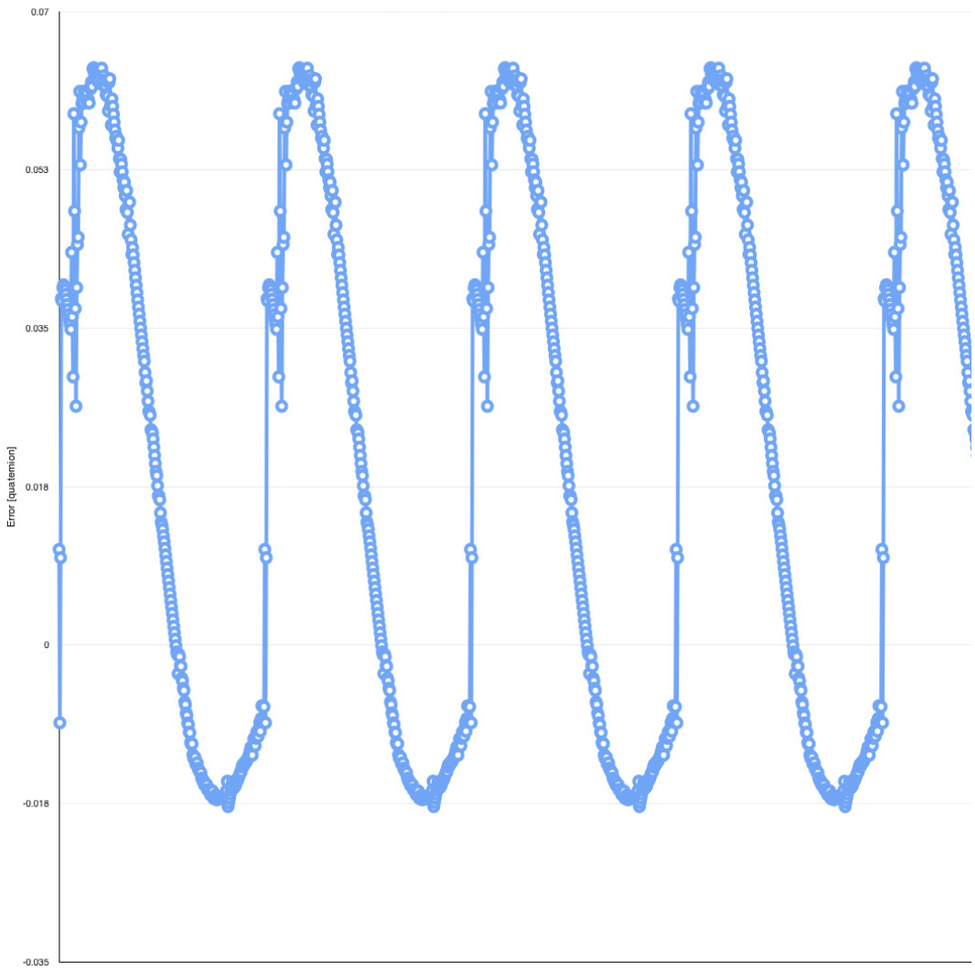


Figure 7.11: Fixed Base Position error, kp = 350

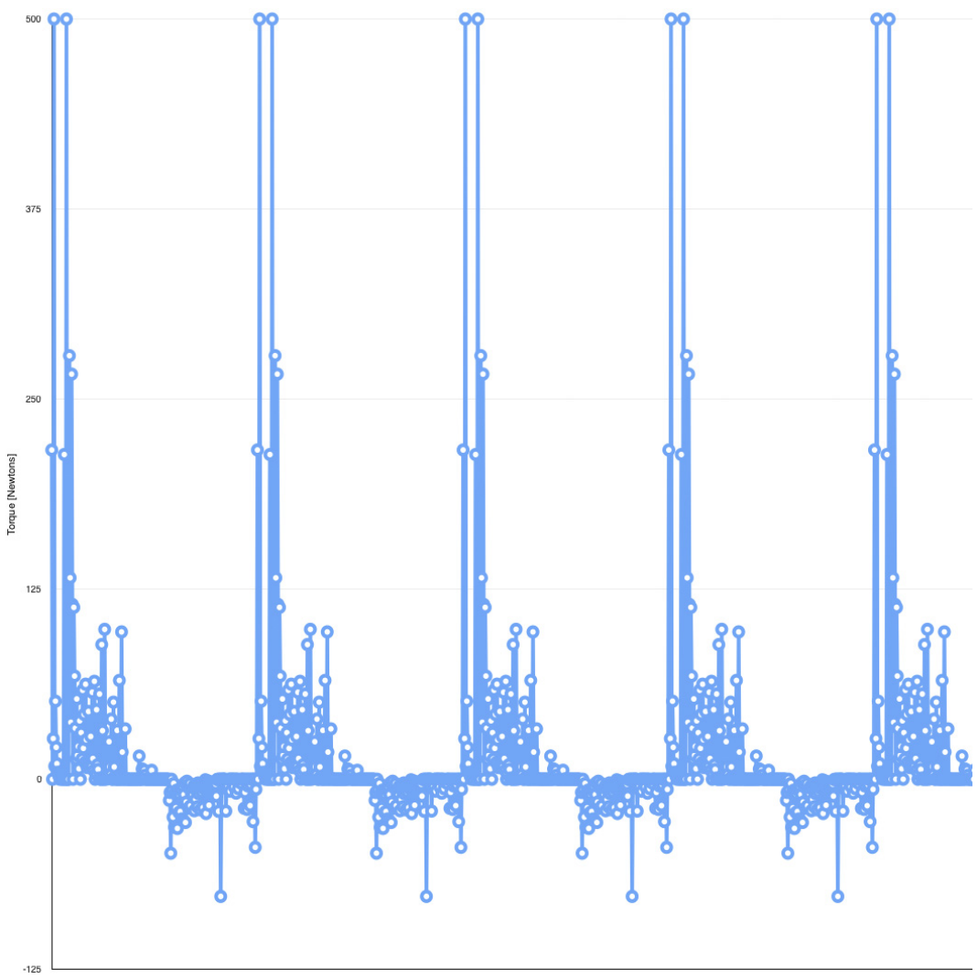


Figure 7.12: Fixed Base Torque, kp = 350

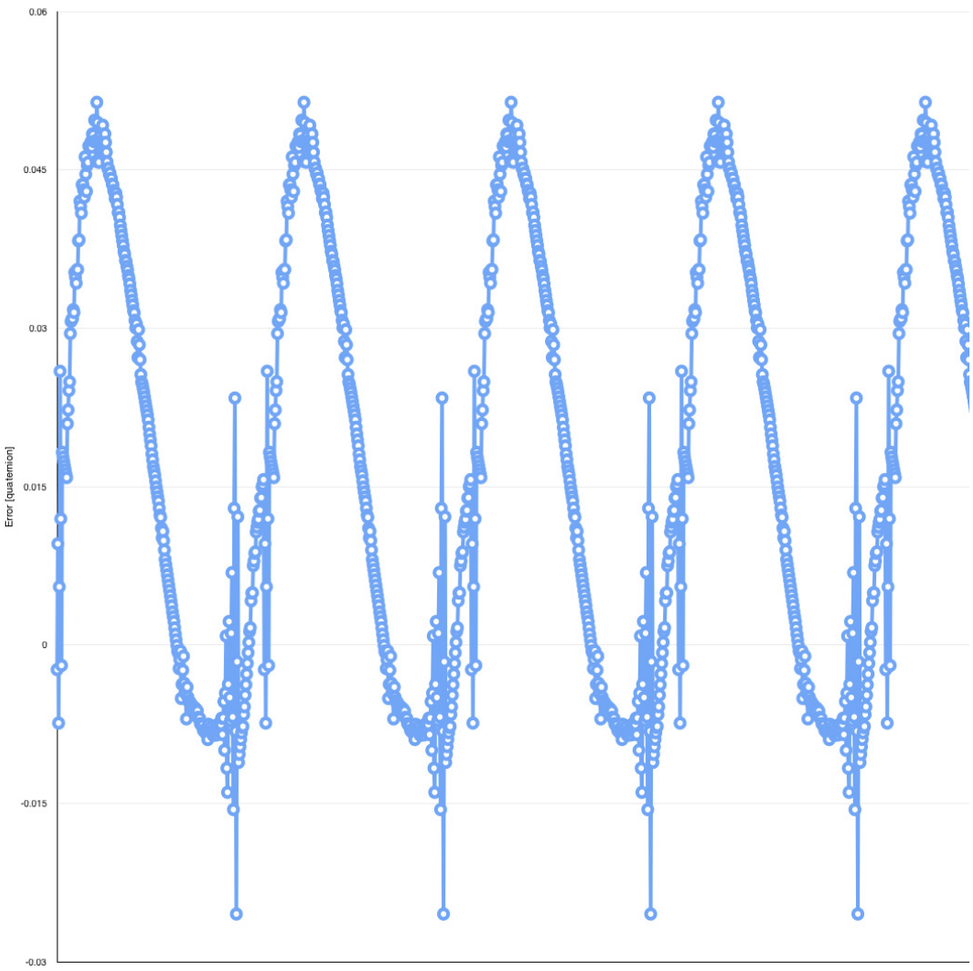


Figure 7.13: Fixed Base Position error, kp = 400

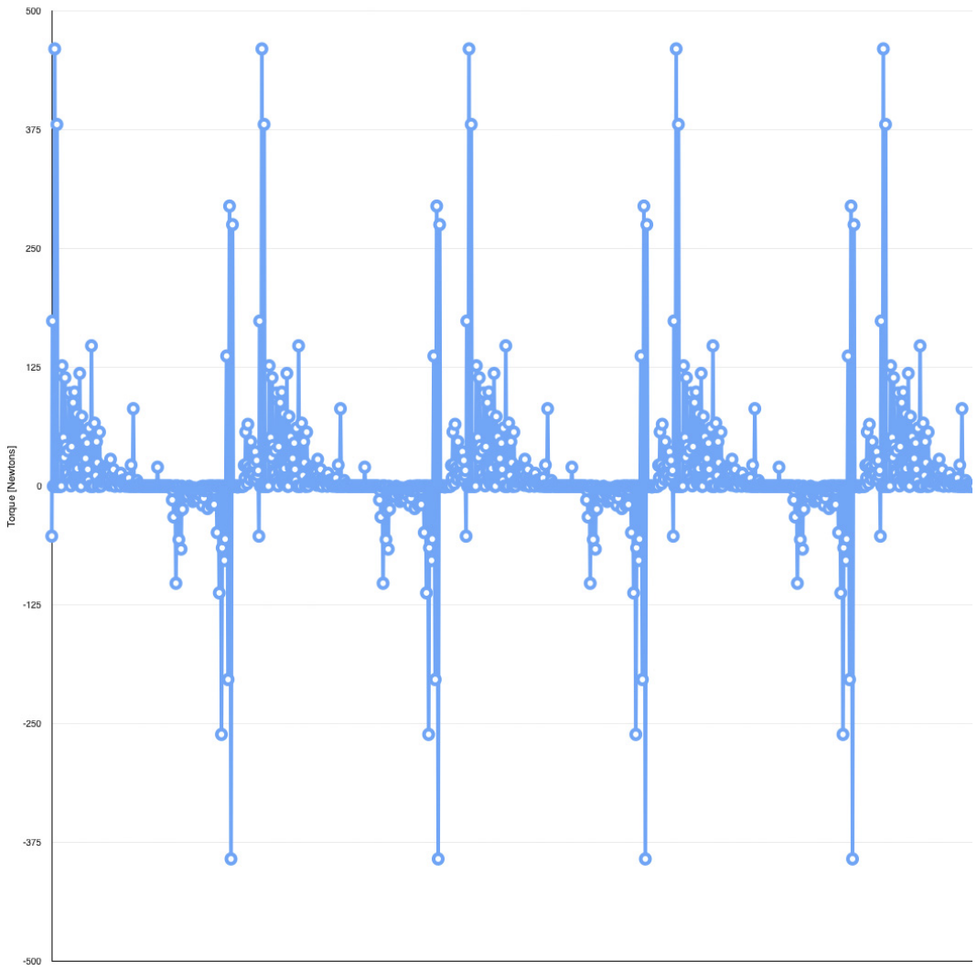


Figure 7.14: Fixed Base Torque, kp = 400

7.0.1.2 Joint 2

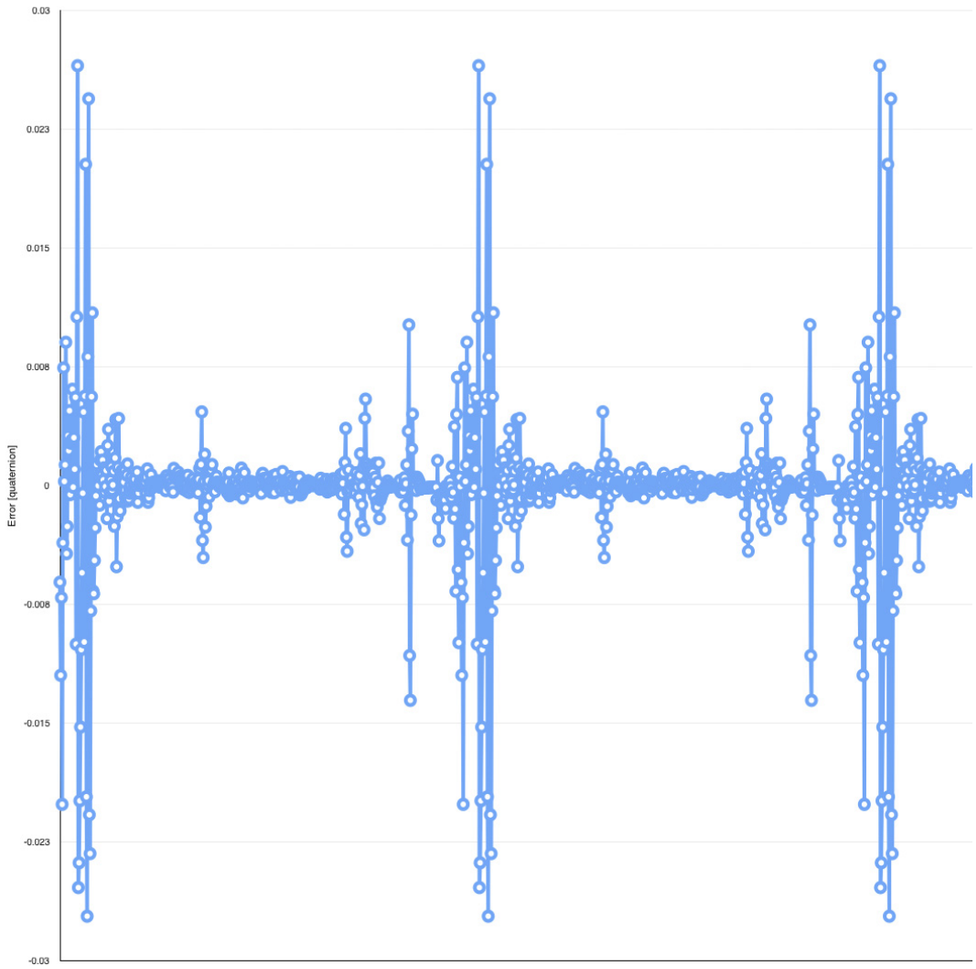


Figure 7.15: Fixed Base Position error, kp = 100

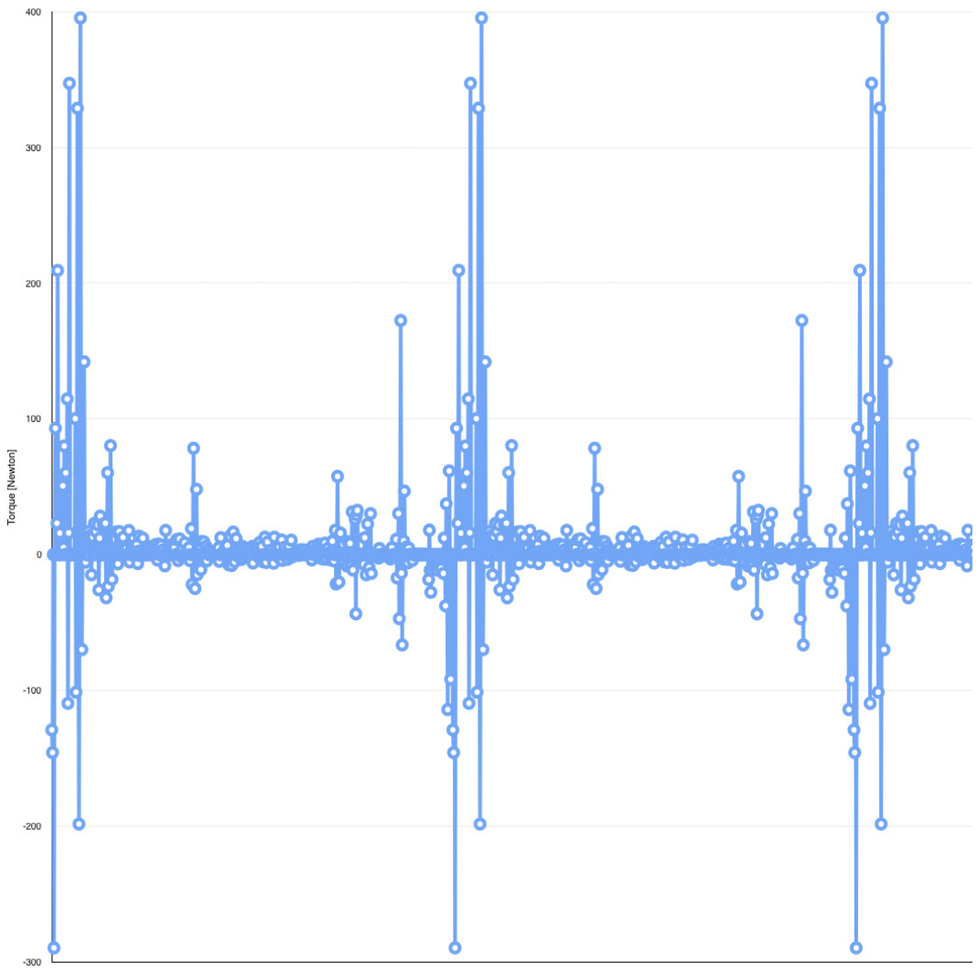


Figure 7.16: Fixed Base Torque, kp = 100

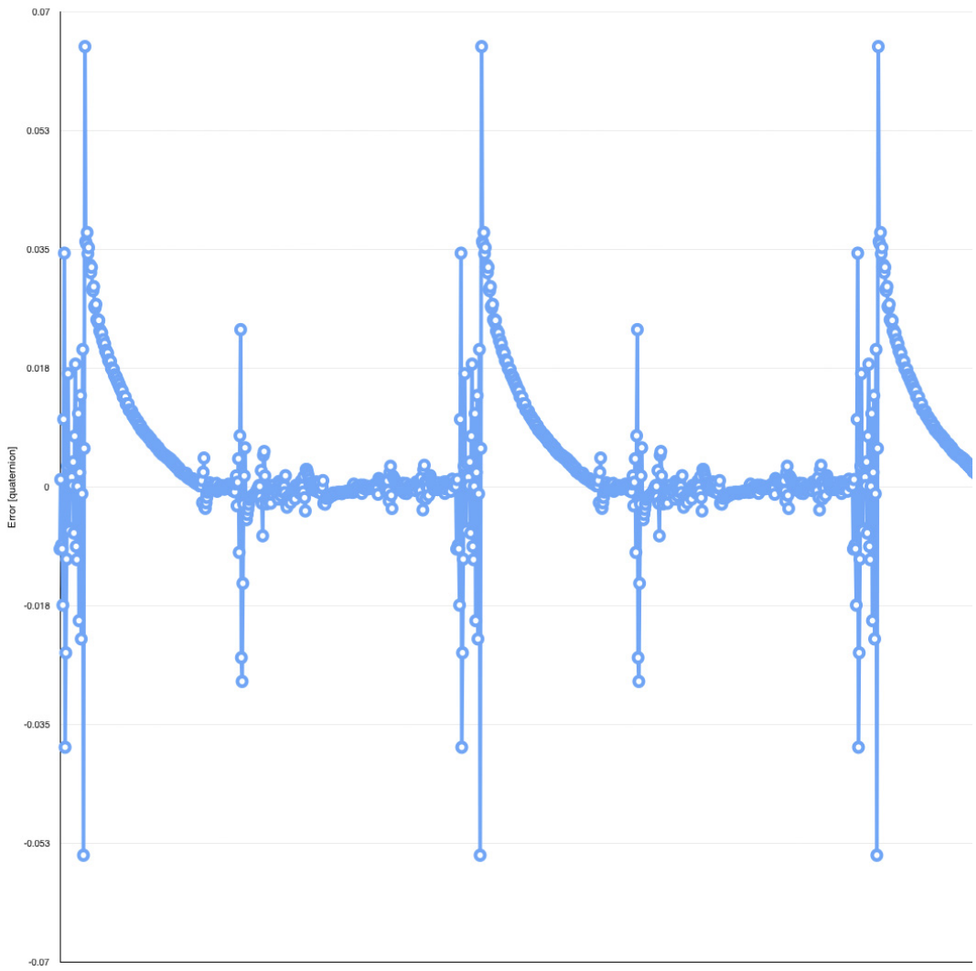


Figure 7.17: Fixed Base Position error, kp = 150

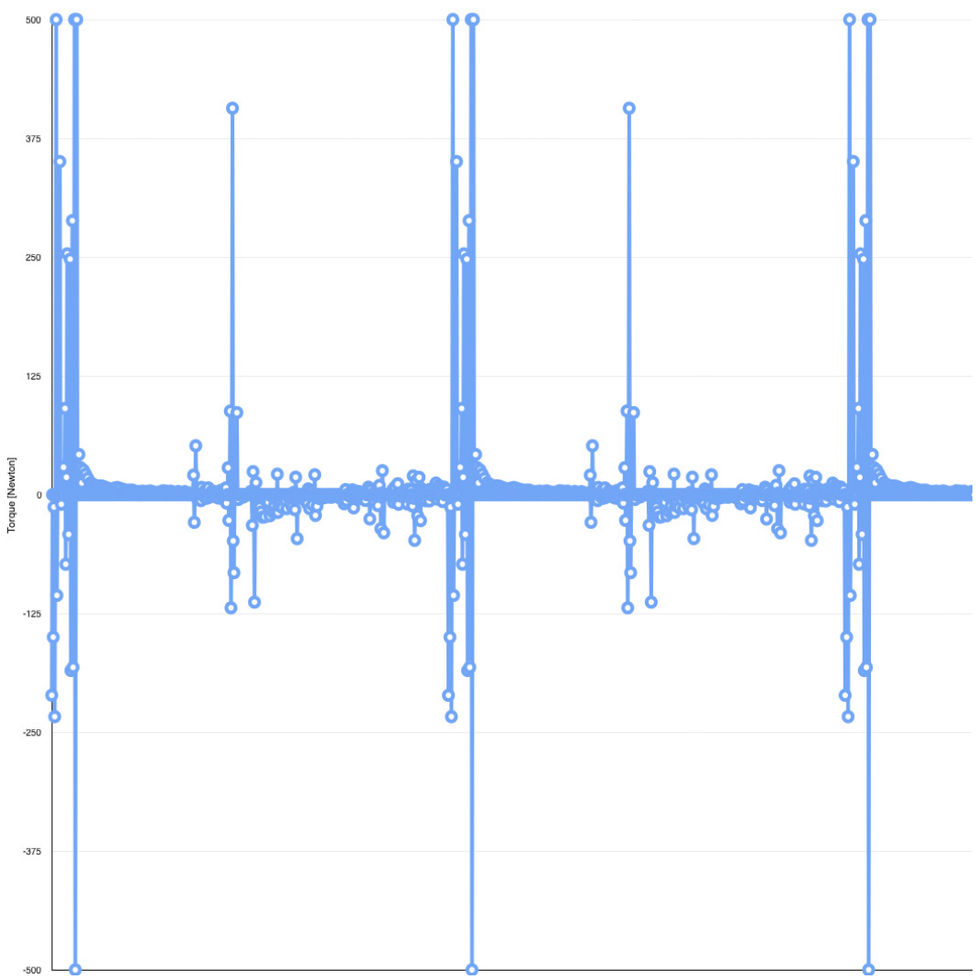


Figure 7.18: Fixed Base Torque, kp = 150

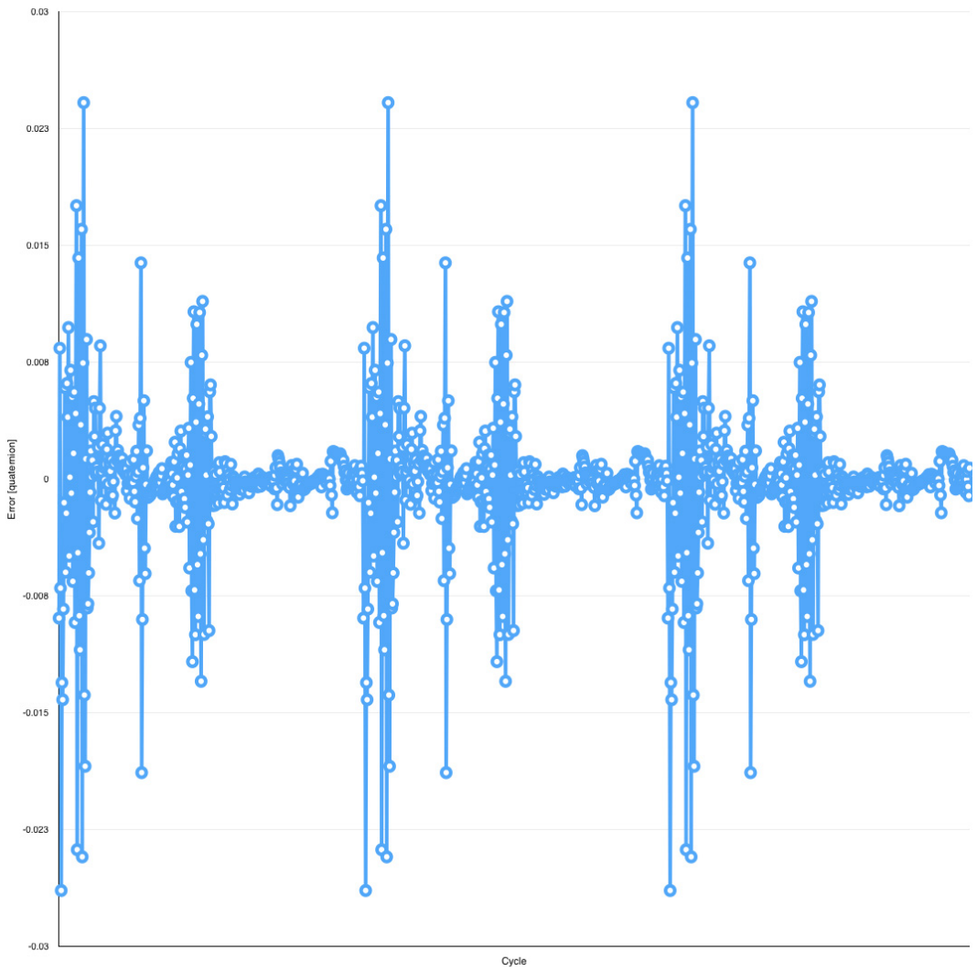


Figure 7.19: Fixed Base Position error, kp = 200

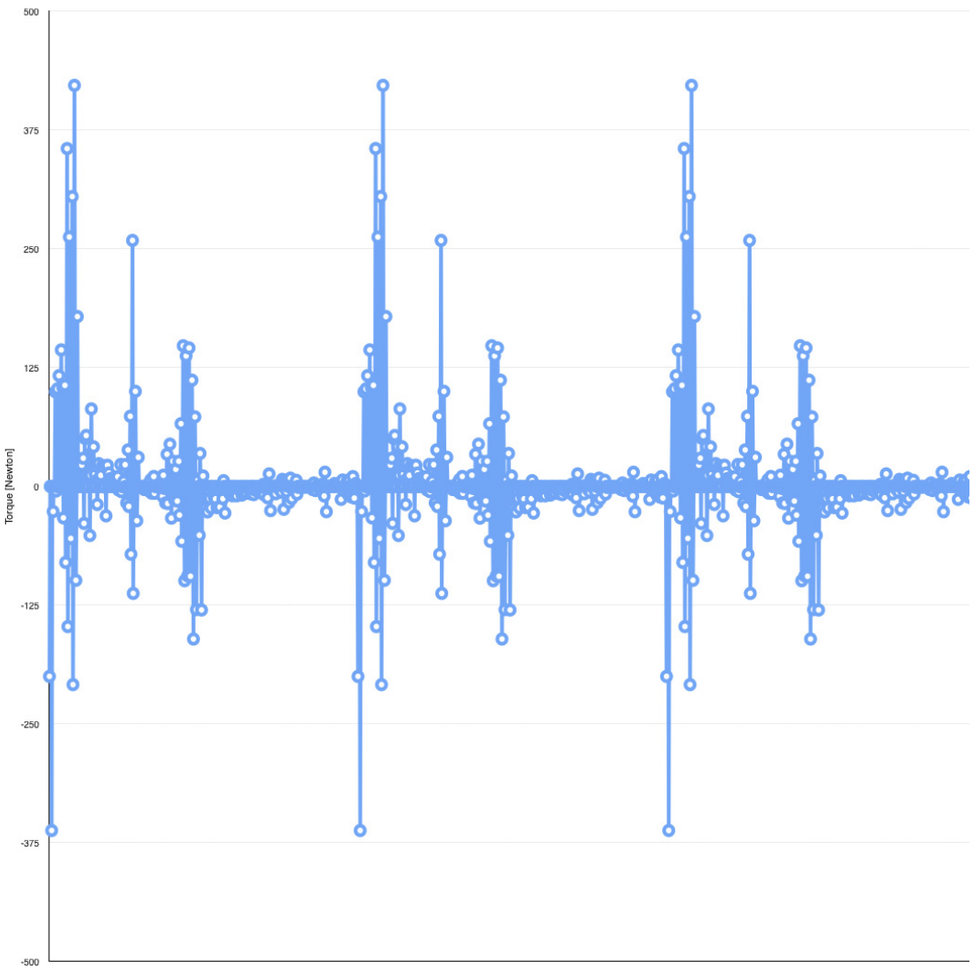


Figure 7.20: Fixed Base Torque, kp = 200

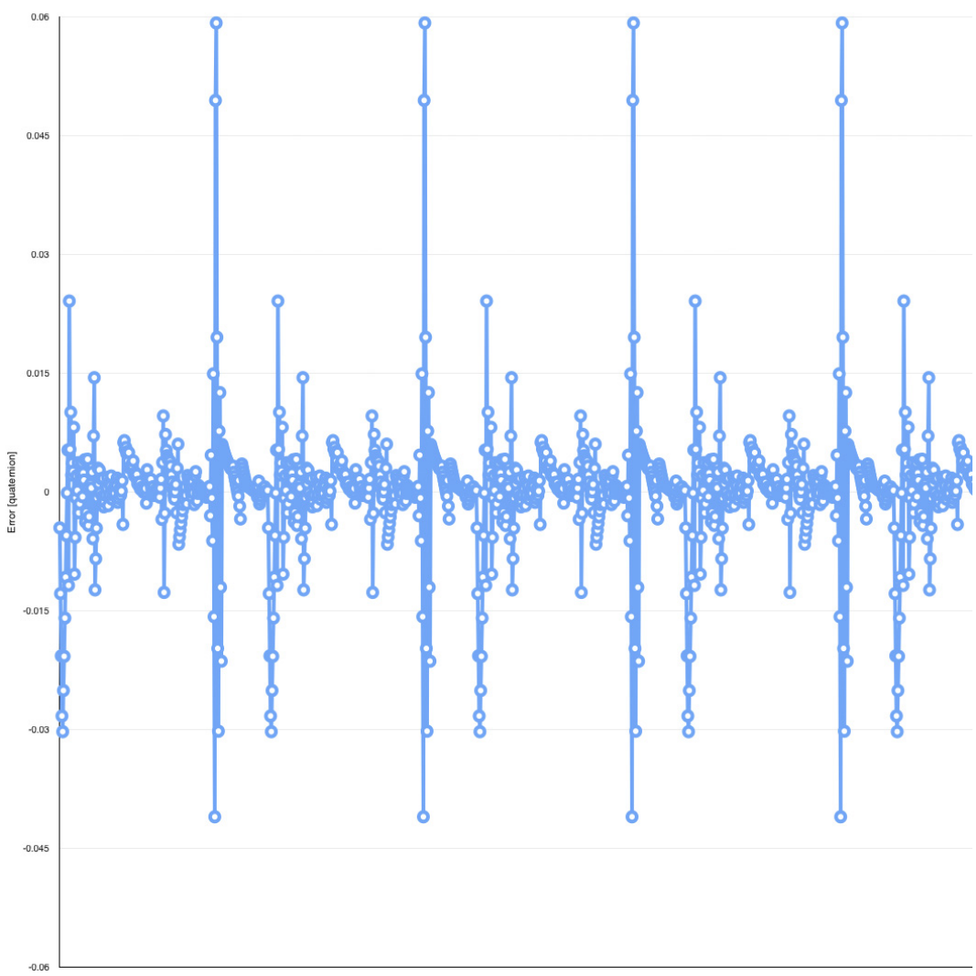


Figure 7.21: Fixed Base Position error, kp = 250

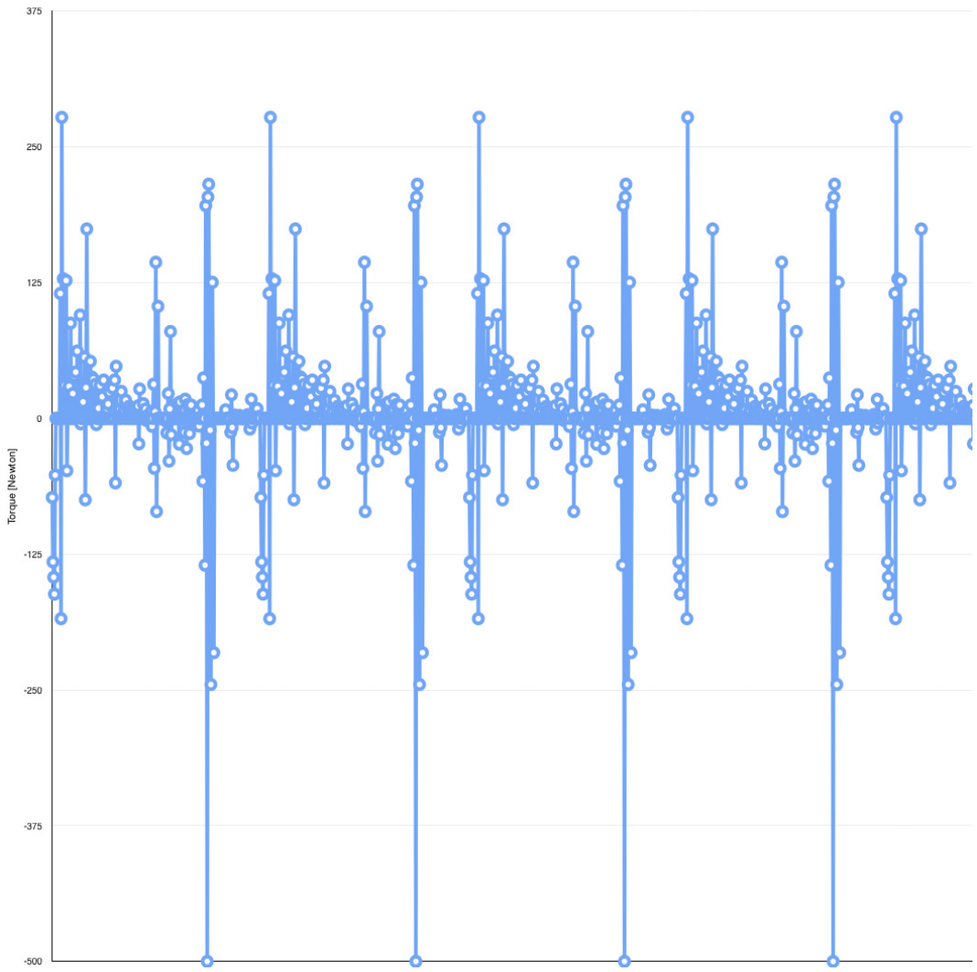


Figure 7.22: Fixed Base Torque, kp = 250

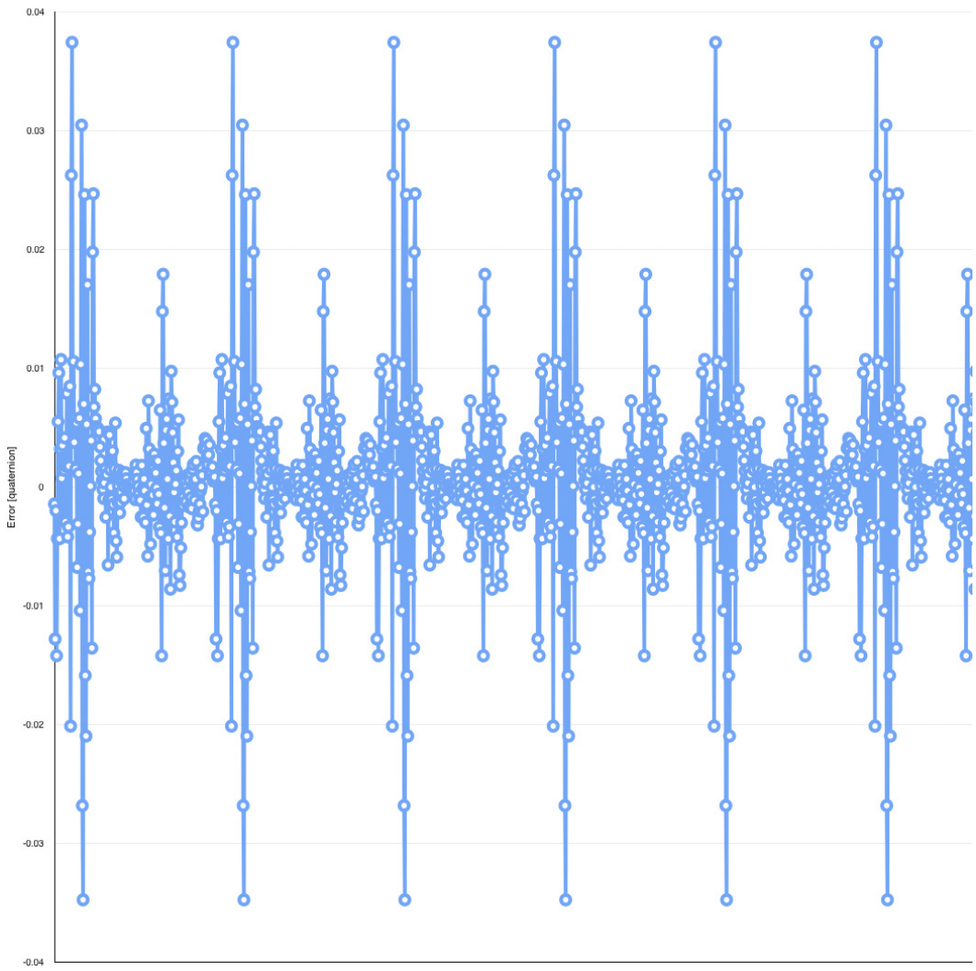


Figure 7.23: Fixed Base Position error, kp = 300

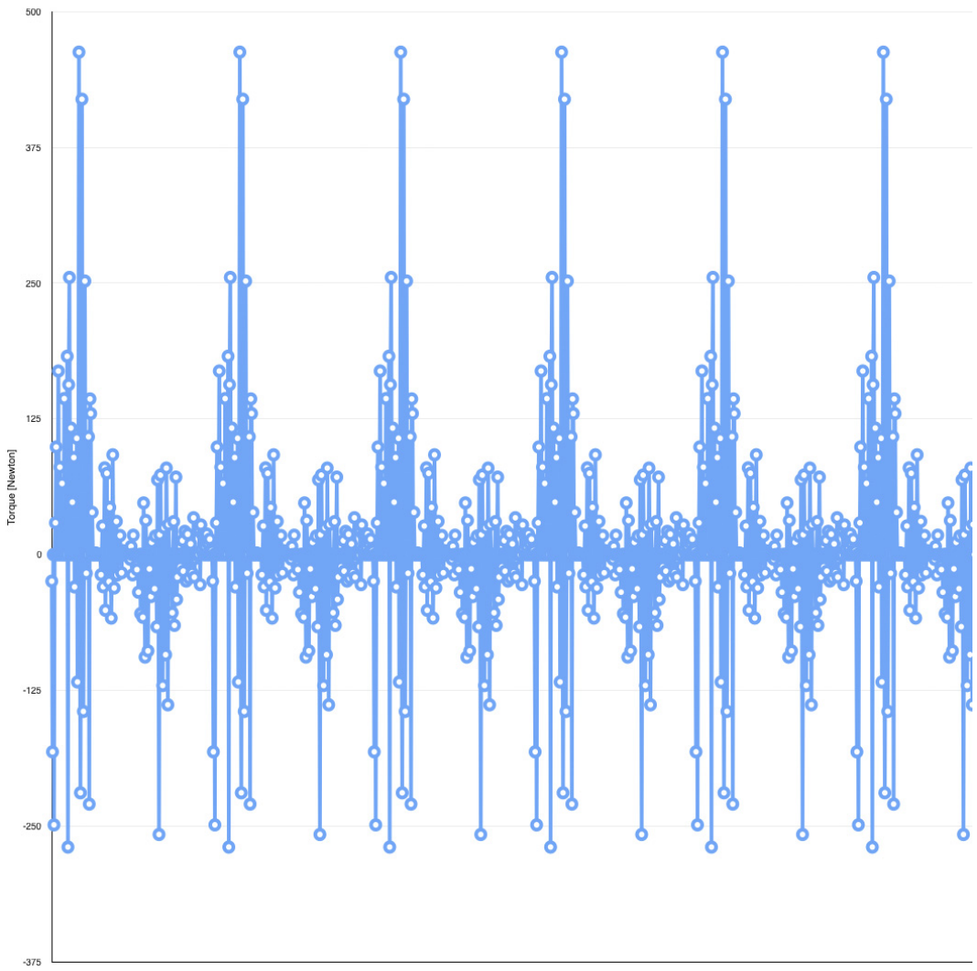


Figure 7.24: Fixed Base Torque, kp = 300

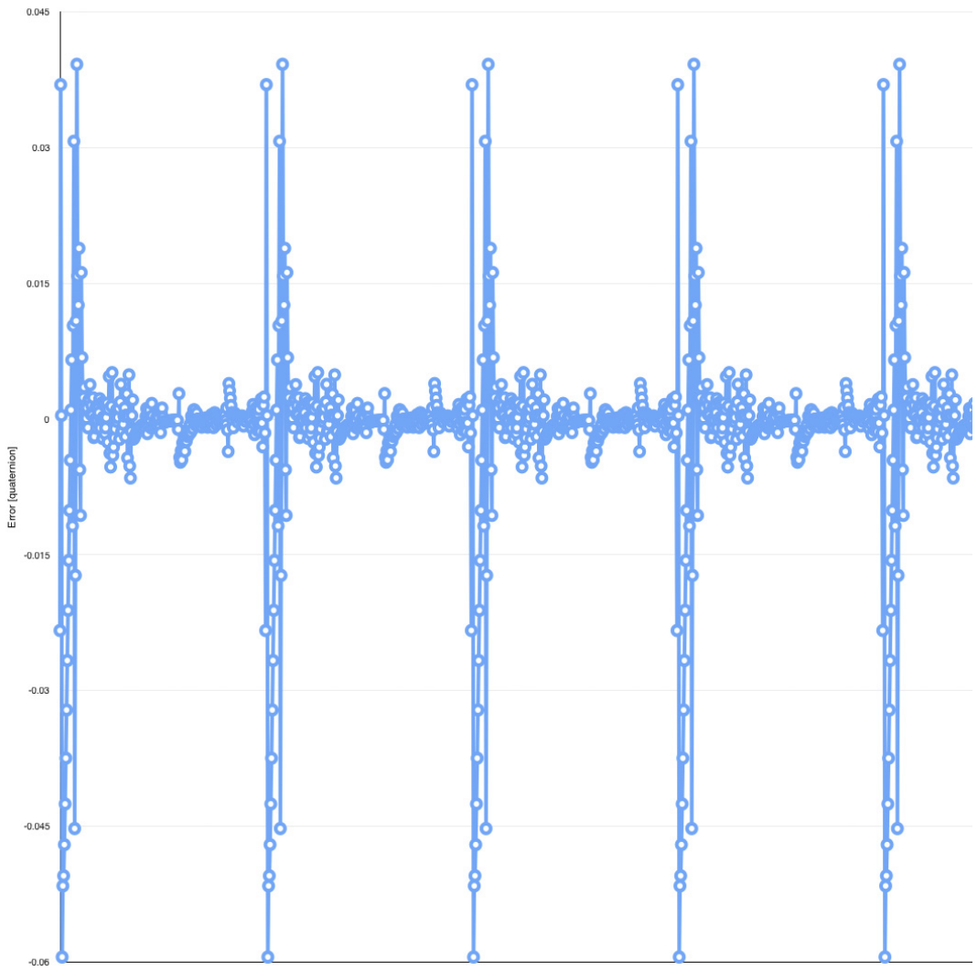


Figure 7.25: Fixed Base Position error, kp = 350

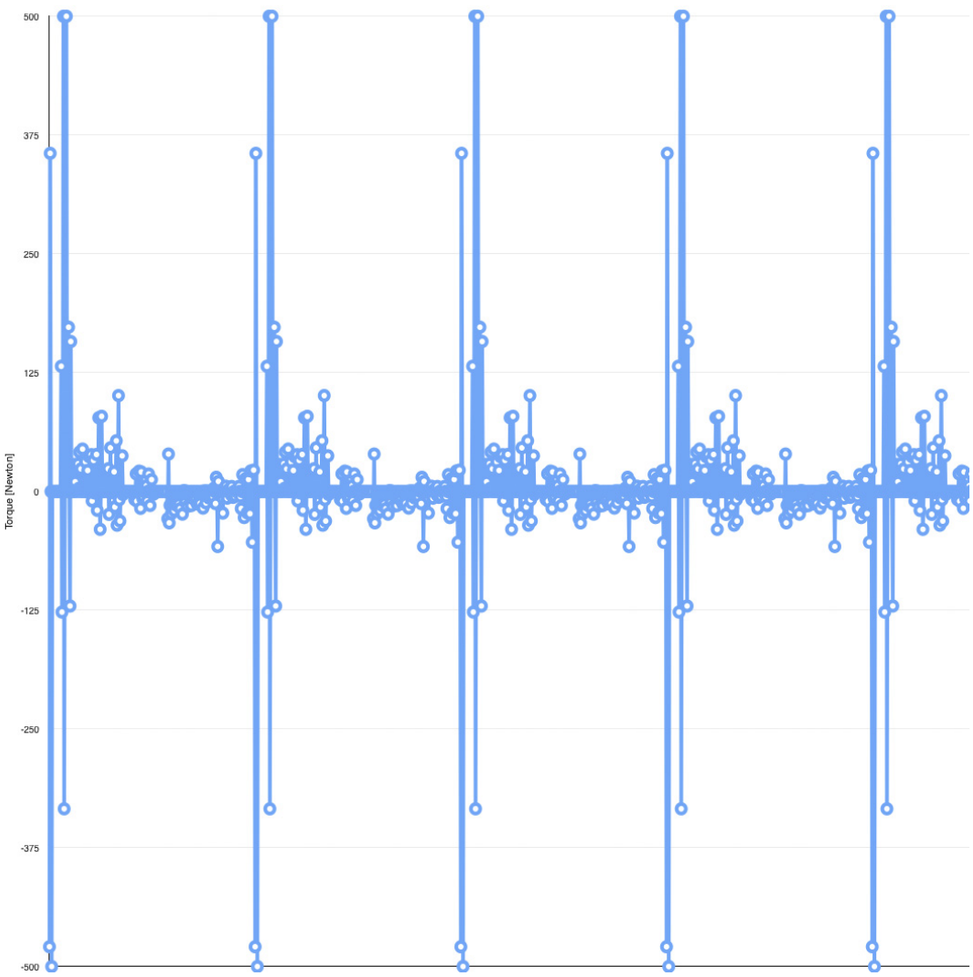


Figure 7.26: Fixed Base Torque, kp = 350

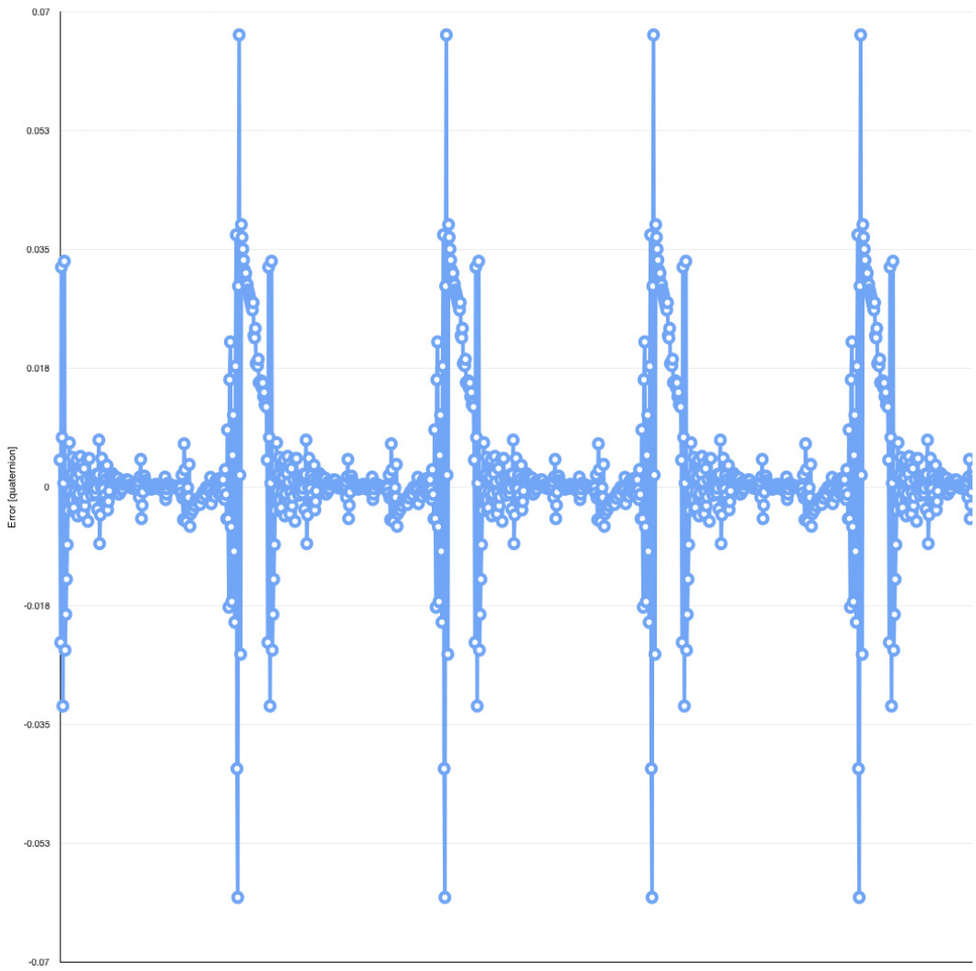


Figure 7.27: Fixed Base Position error, kp = 400

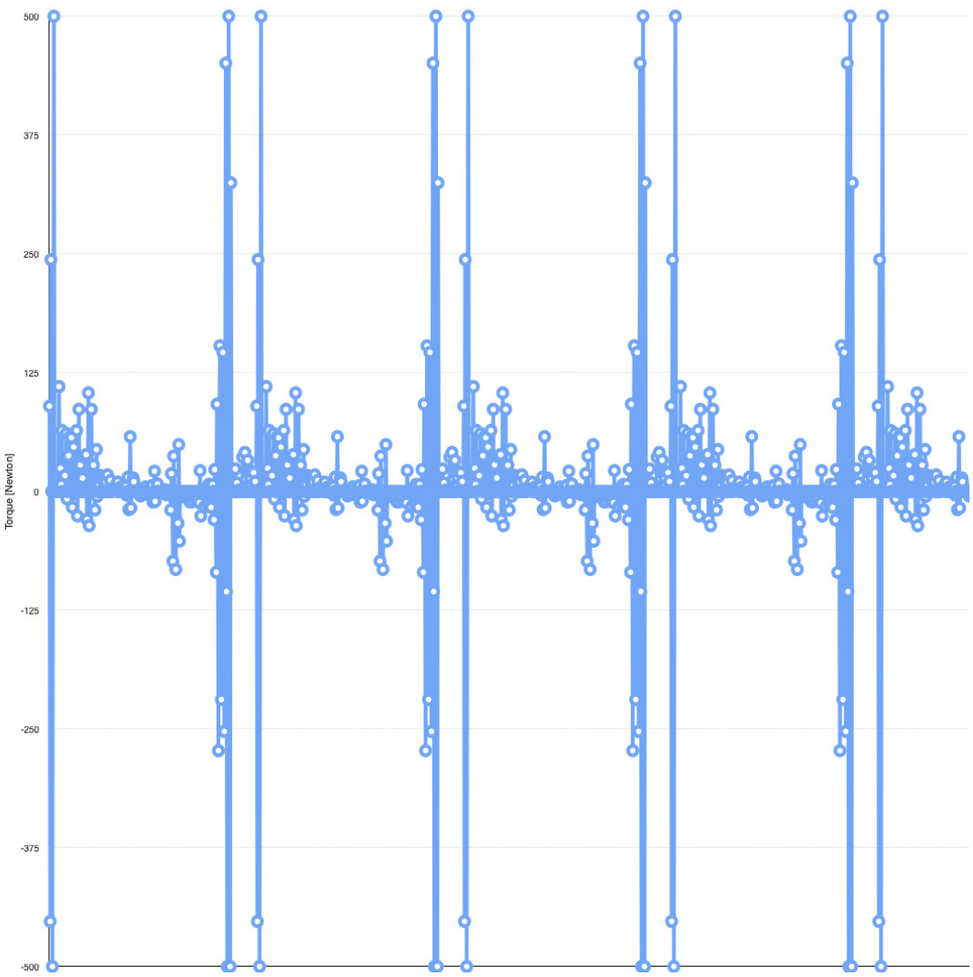


Figure 7.28: Fixed Base Torque, kp = 400

7.0.2 Integral variable

7.0.2.1 Joint 1

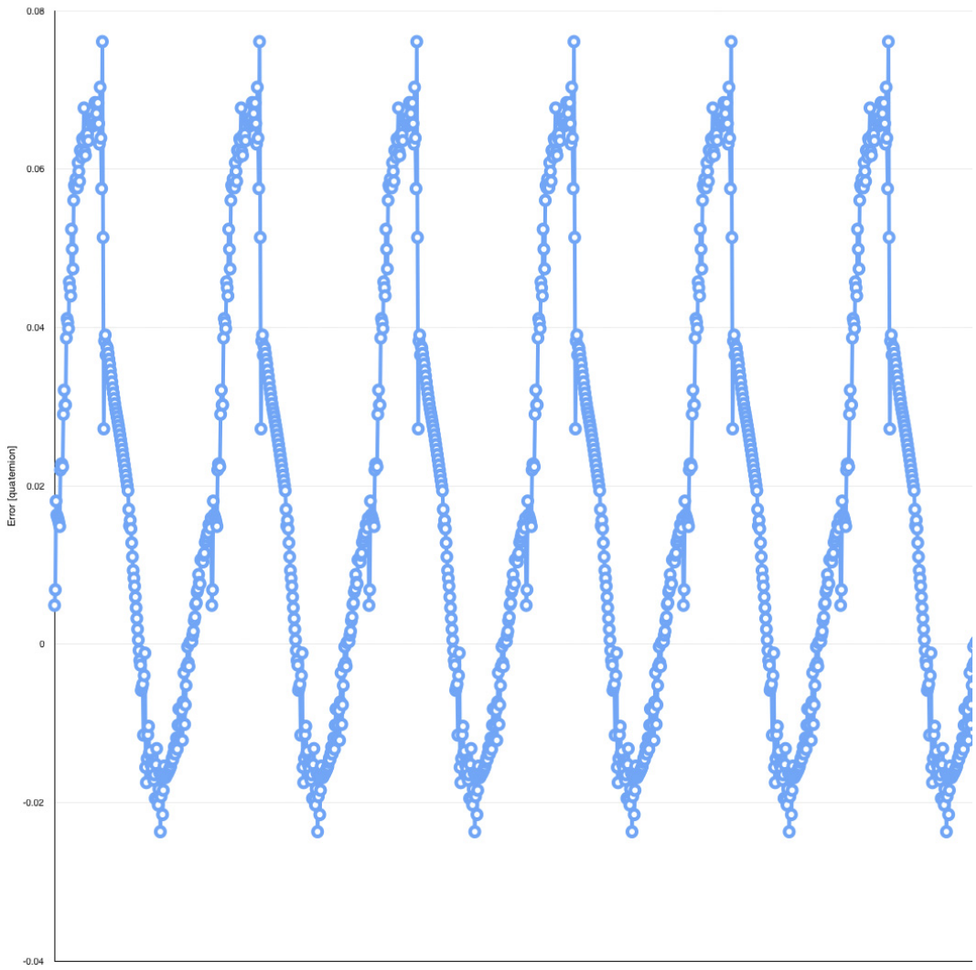


Figure 7.29: Fixed Base Position error, Ki = 100

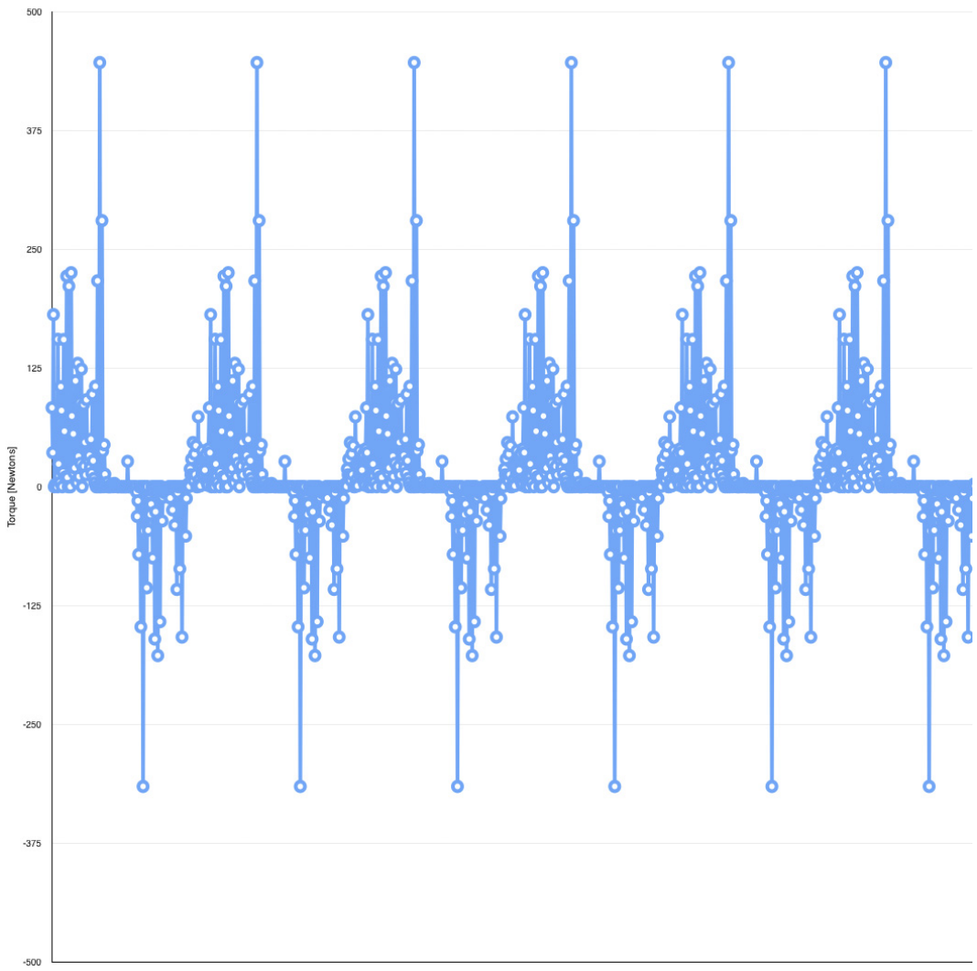


Figure 7.30: Fixed Base Torque, Ki = 100

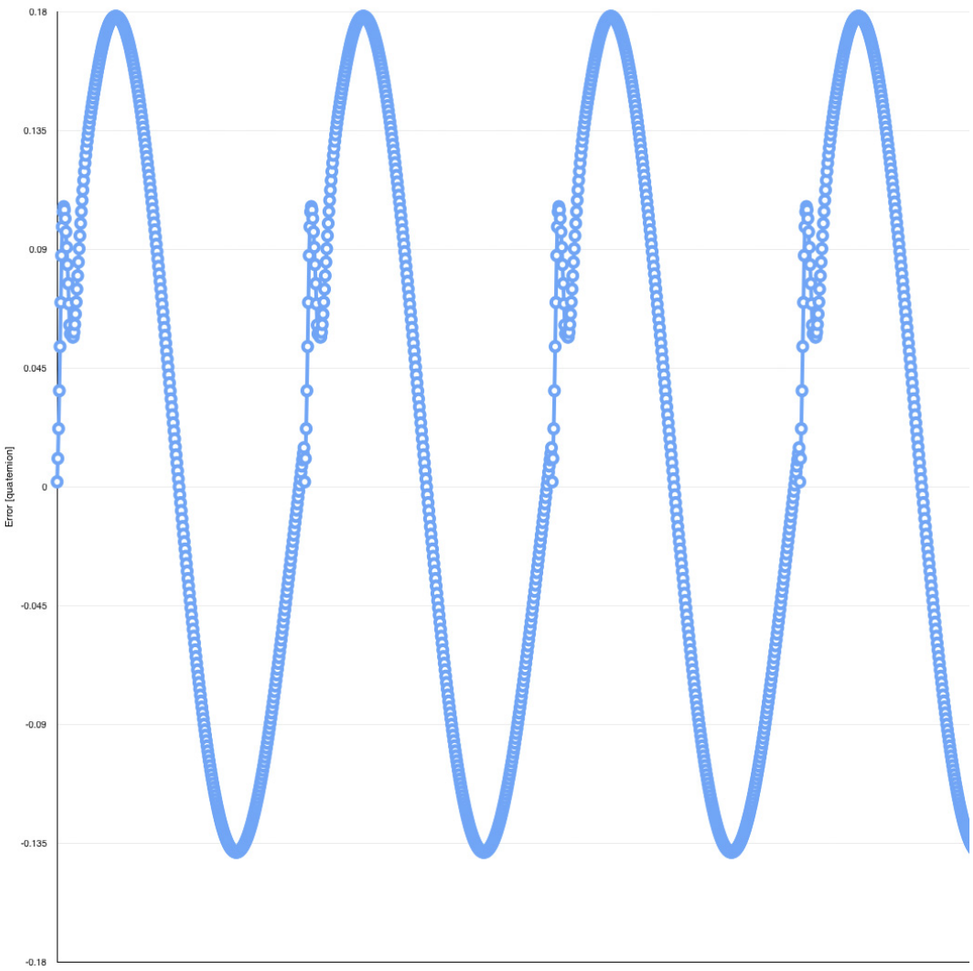


Figure 7.31: Fixed Base Position error, Ki = 150

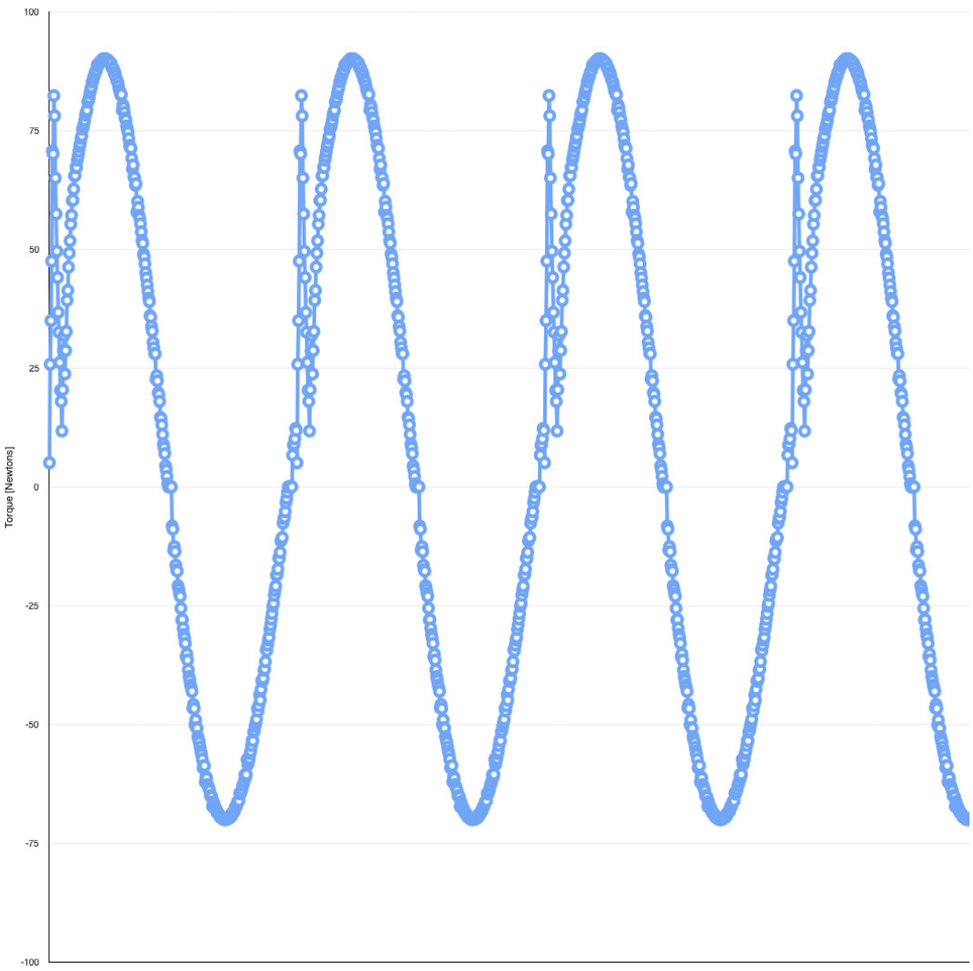


Figure 7.32: Fixed Base Torque, Ki = 150

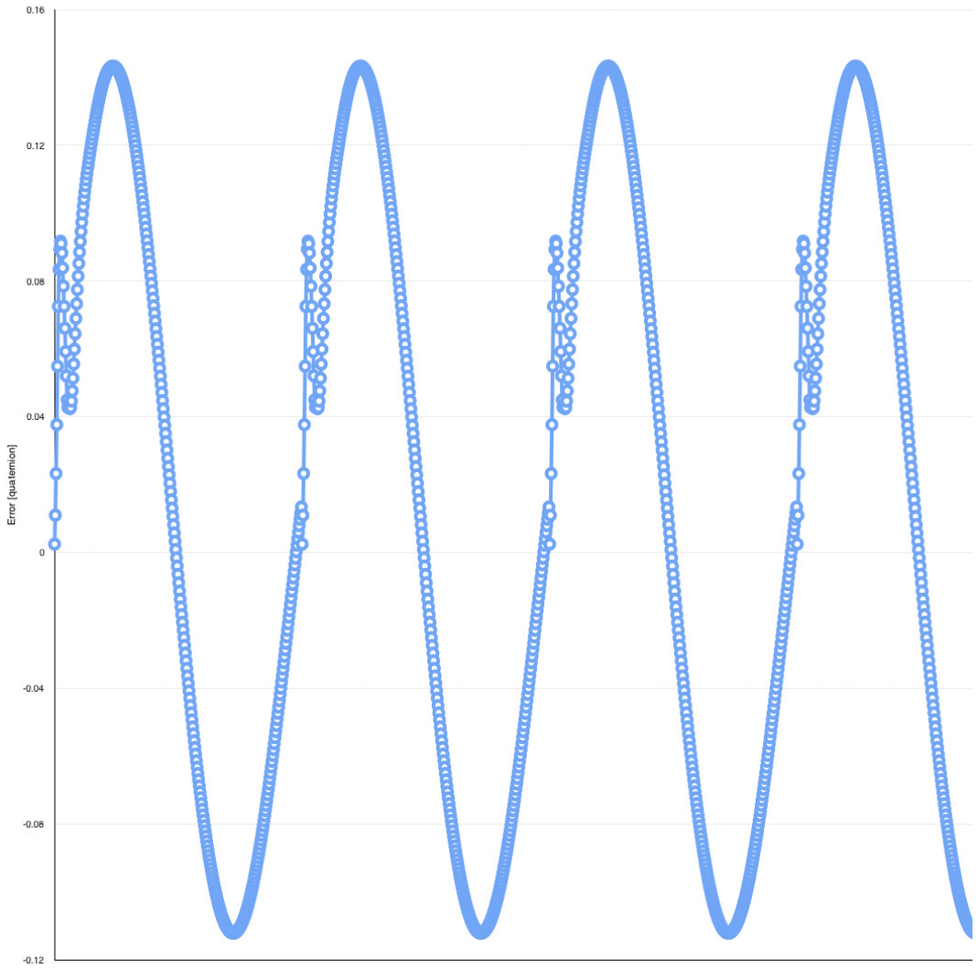


Figure 7.33: Fixed Base Position error, Ki = 200

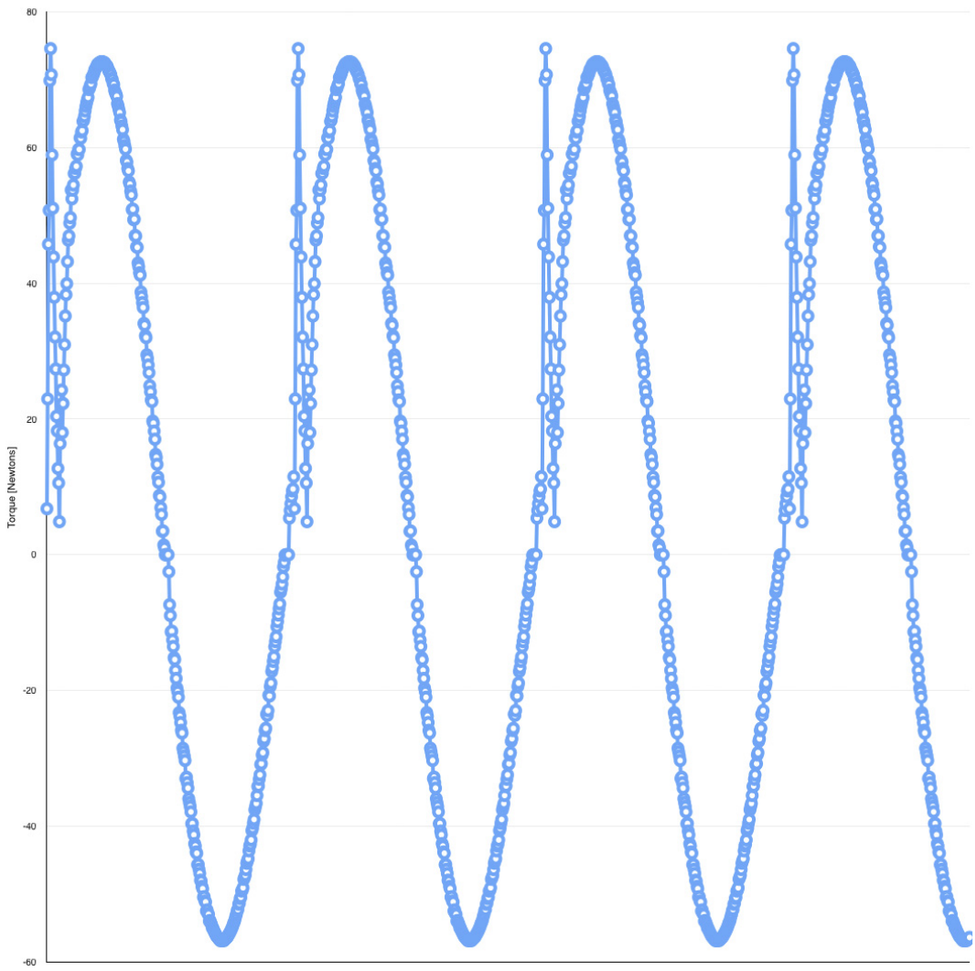


Figure 7.34: Fixed Base Torque, Ki = 200

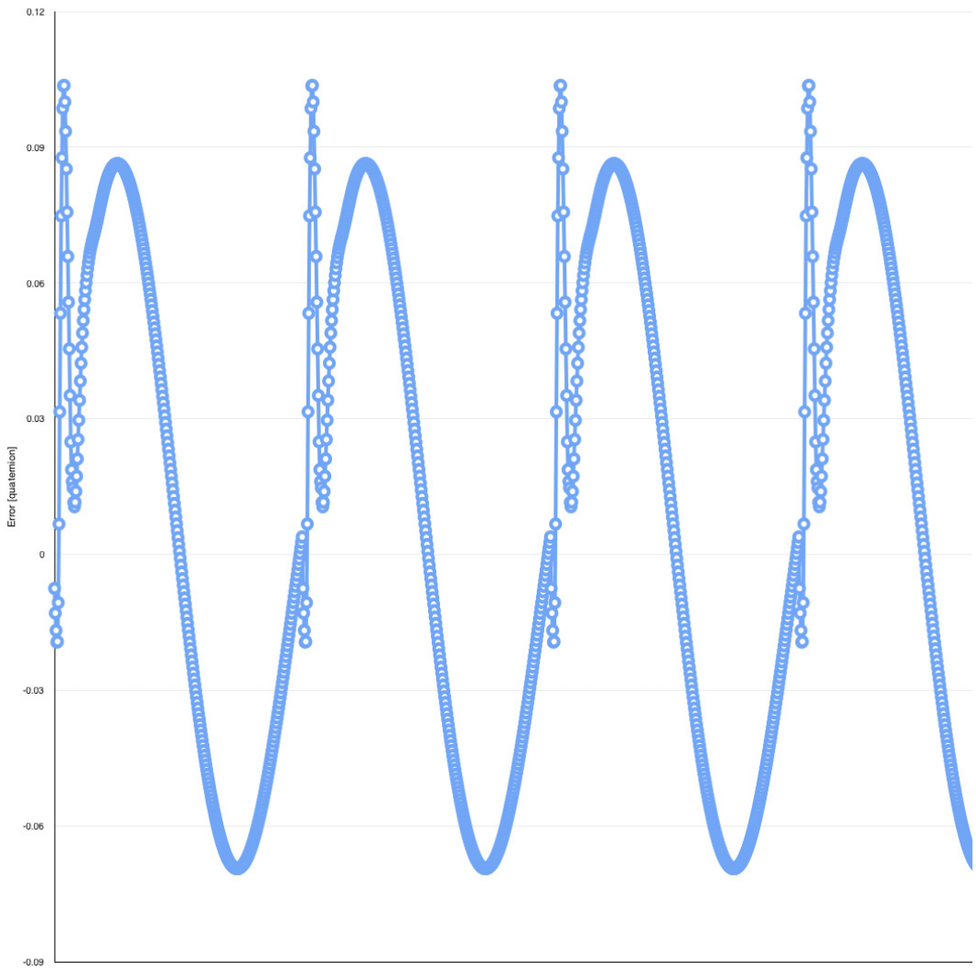


Figure 7.35: Fixed Base Position error, Ki = 250

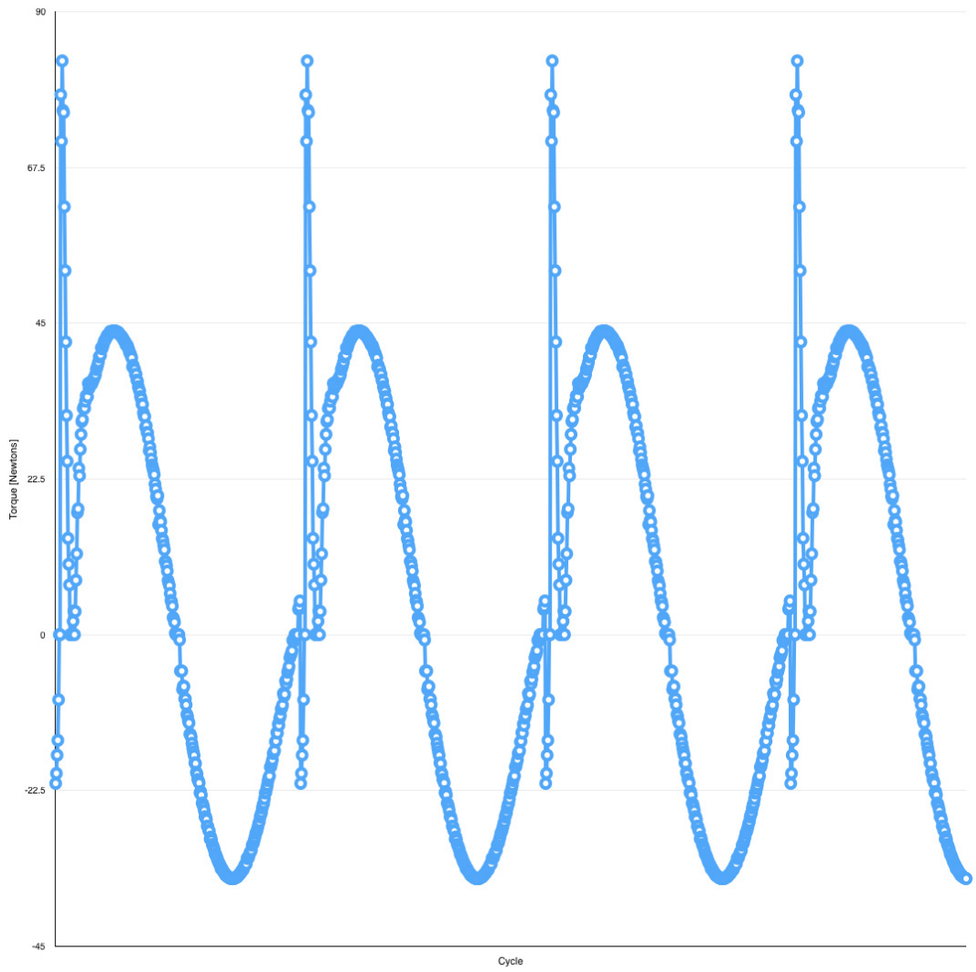


Figure 7.36: Fixed Base Torque, Ki = 250

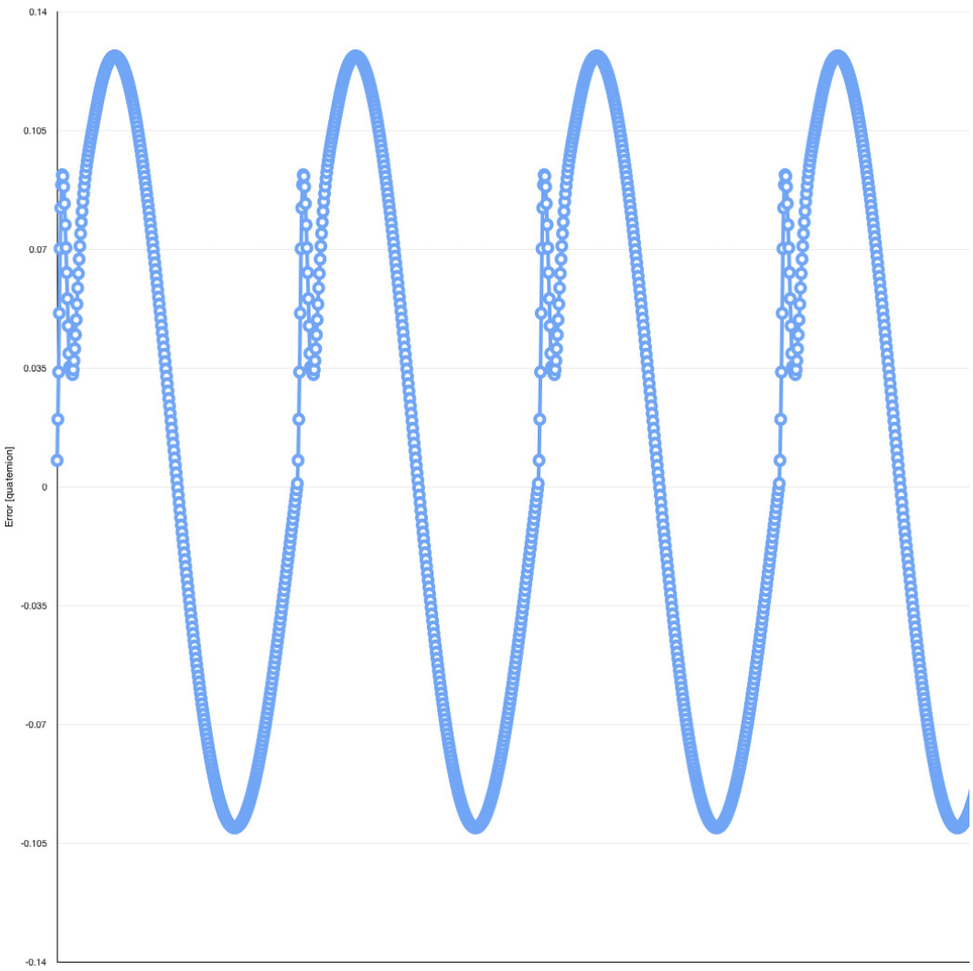


Figure 7.37: Fixed Base Position error, Ki = 300

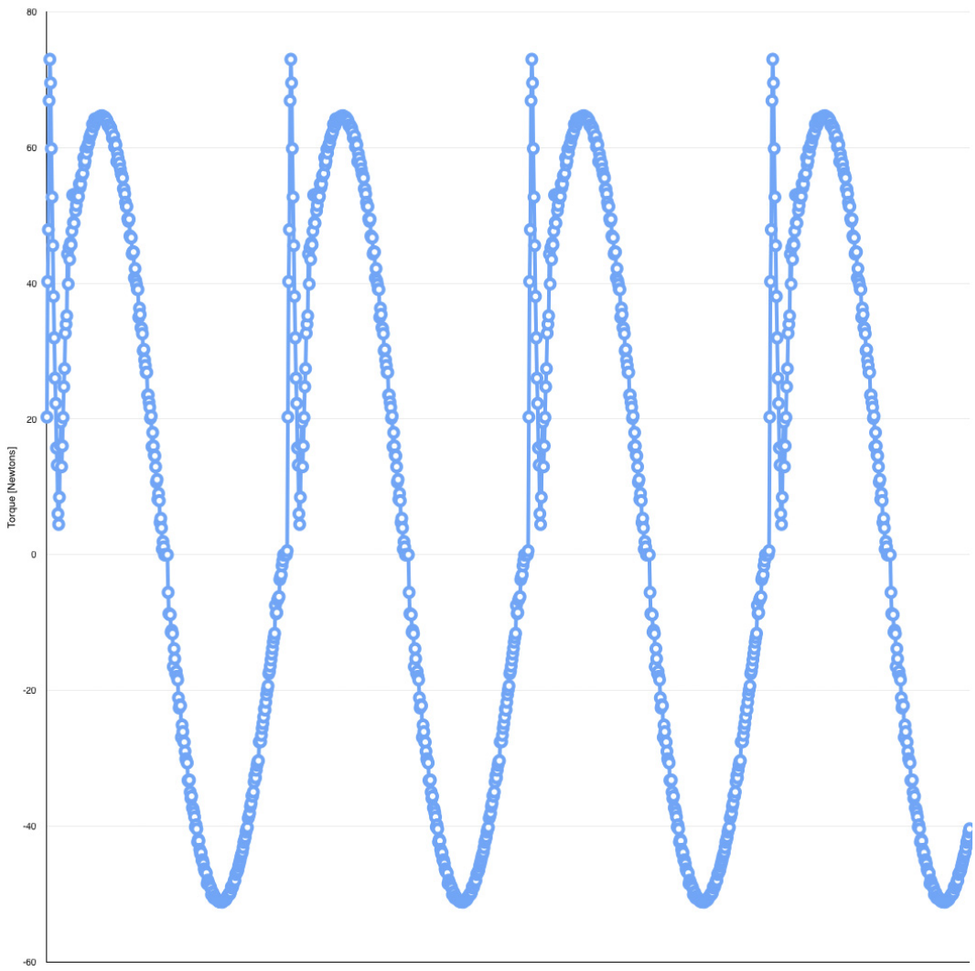


Figure 7.38: Fixed Base Torque, Ki = 300

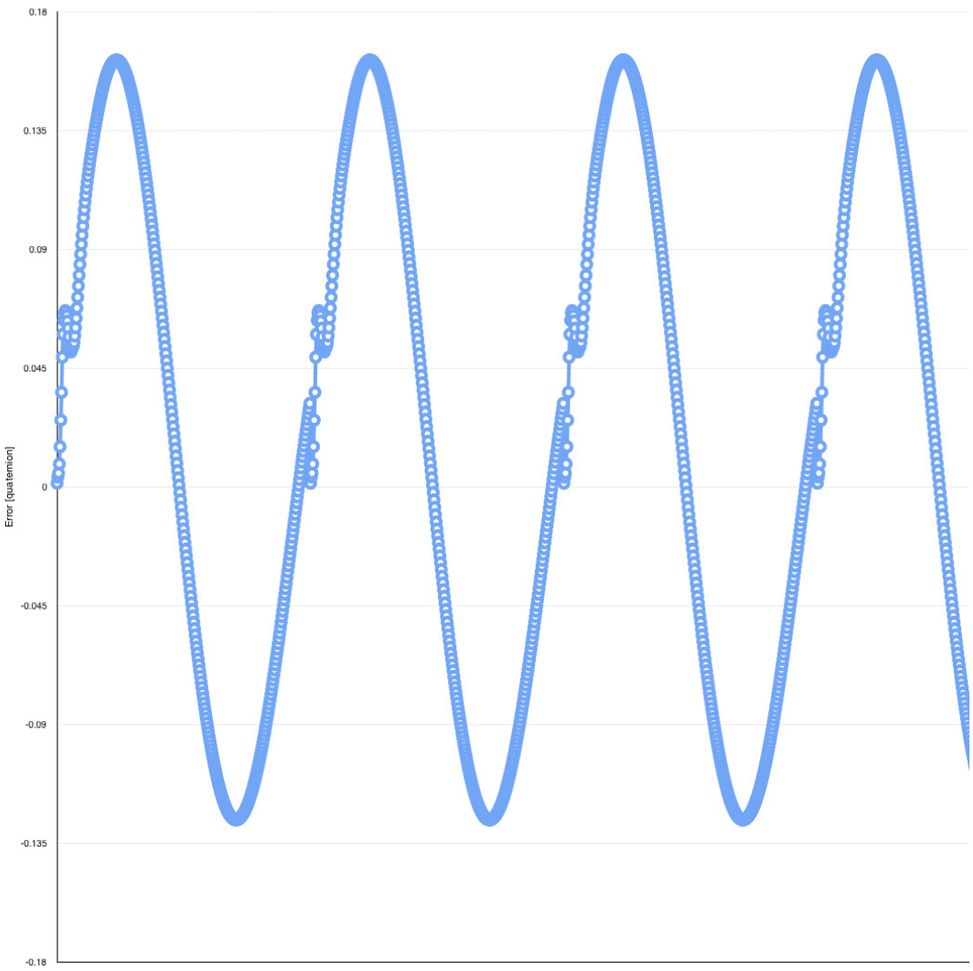


Figure 7.39: Fixed Base Position error, Ki = 350

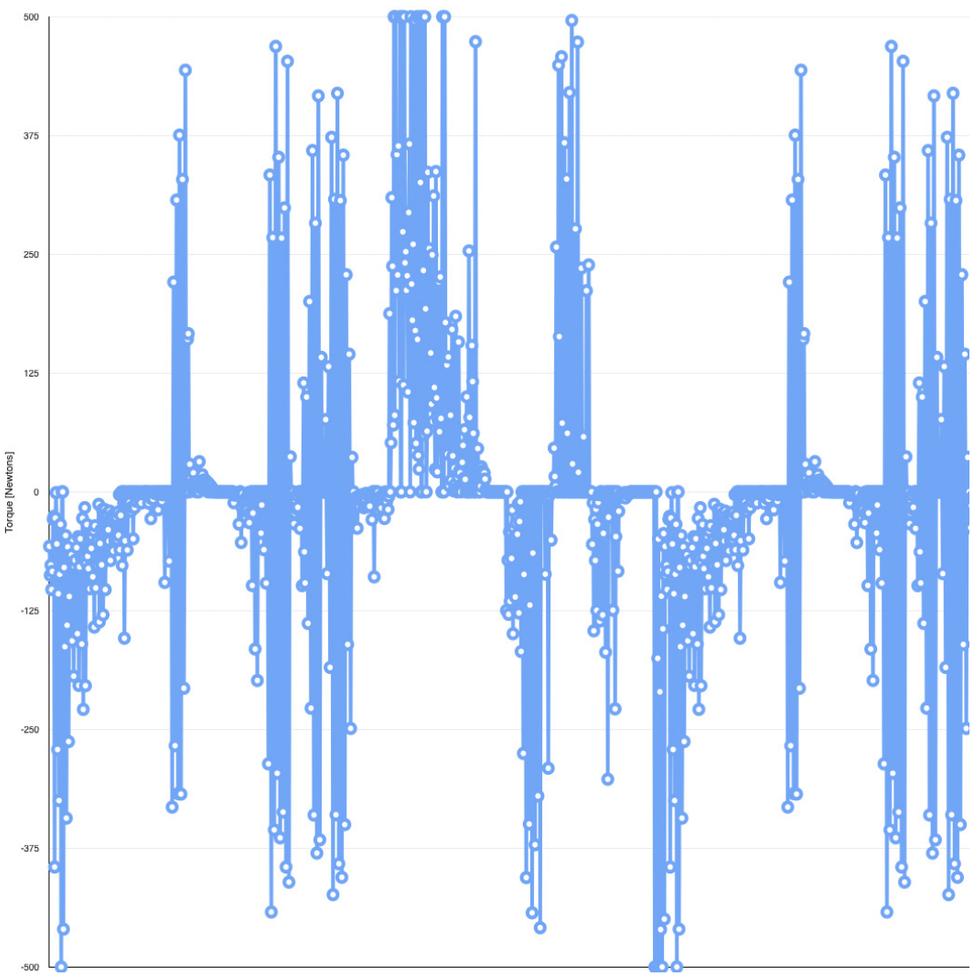


Figure 7.40: Fixed Base Torque, Ki = 350

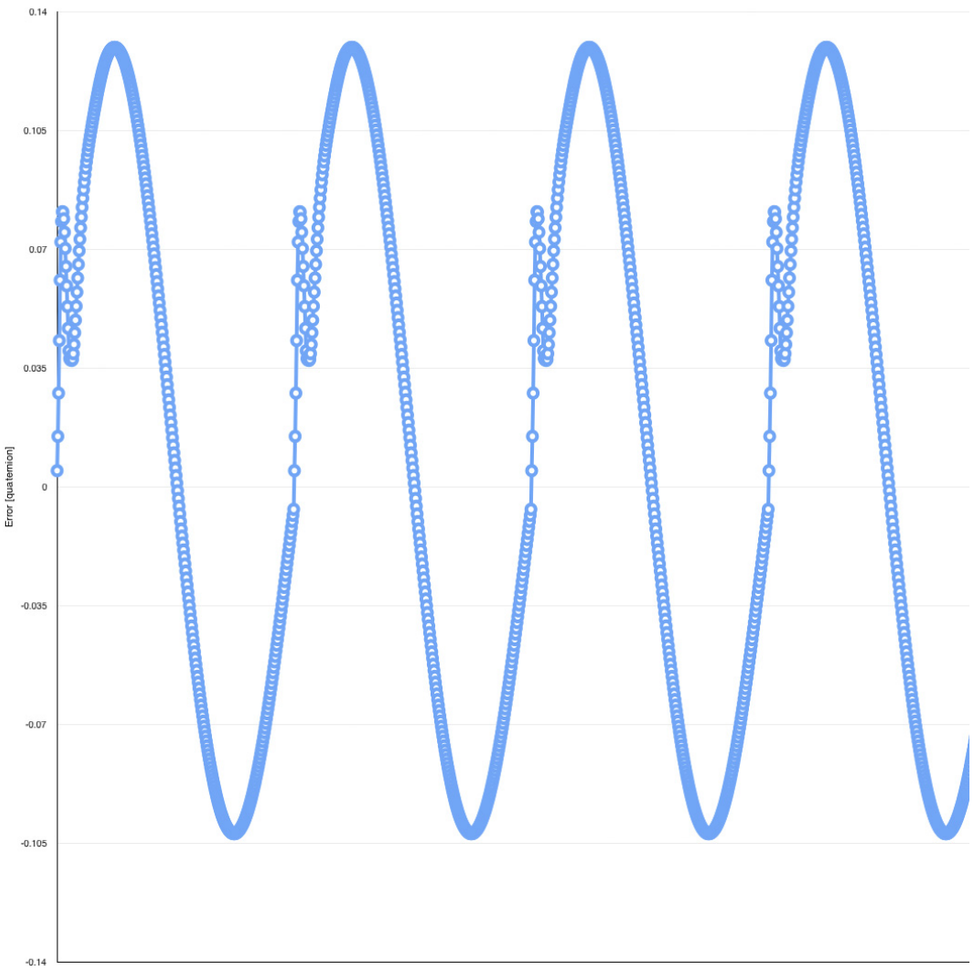


Figure 7.41: Fixed Base Position error, Ki = 400

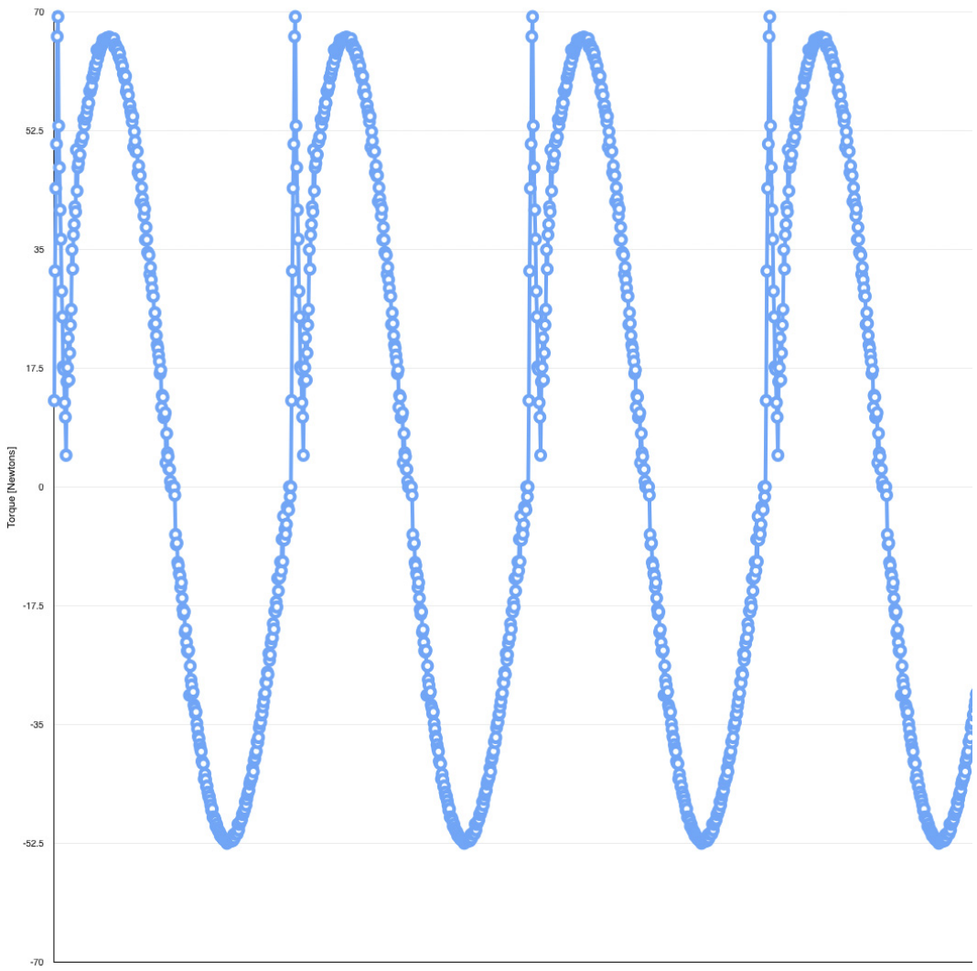


Figure 7.42: Fixed Base Torque, Ki = 400

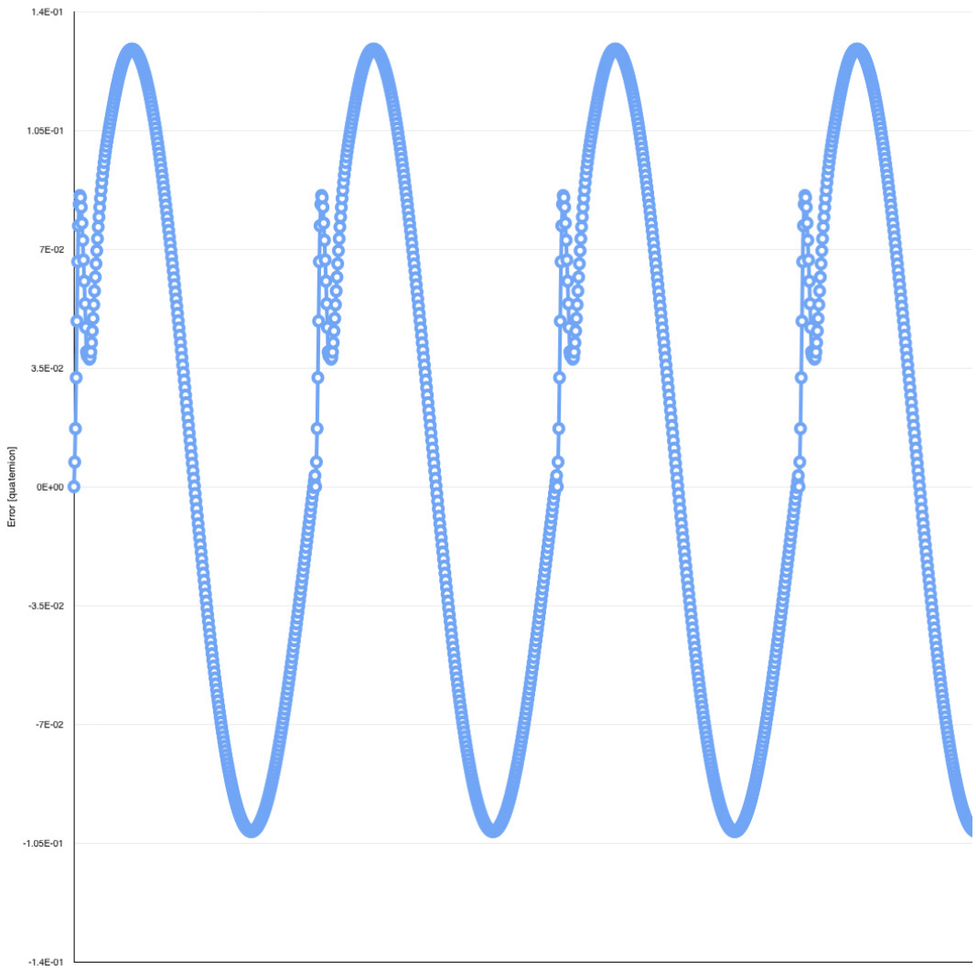


Figure 7.43: Fixed Base Position error, Ki = 450

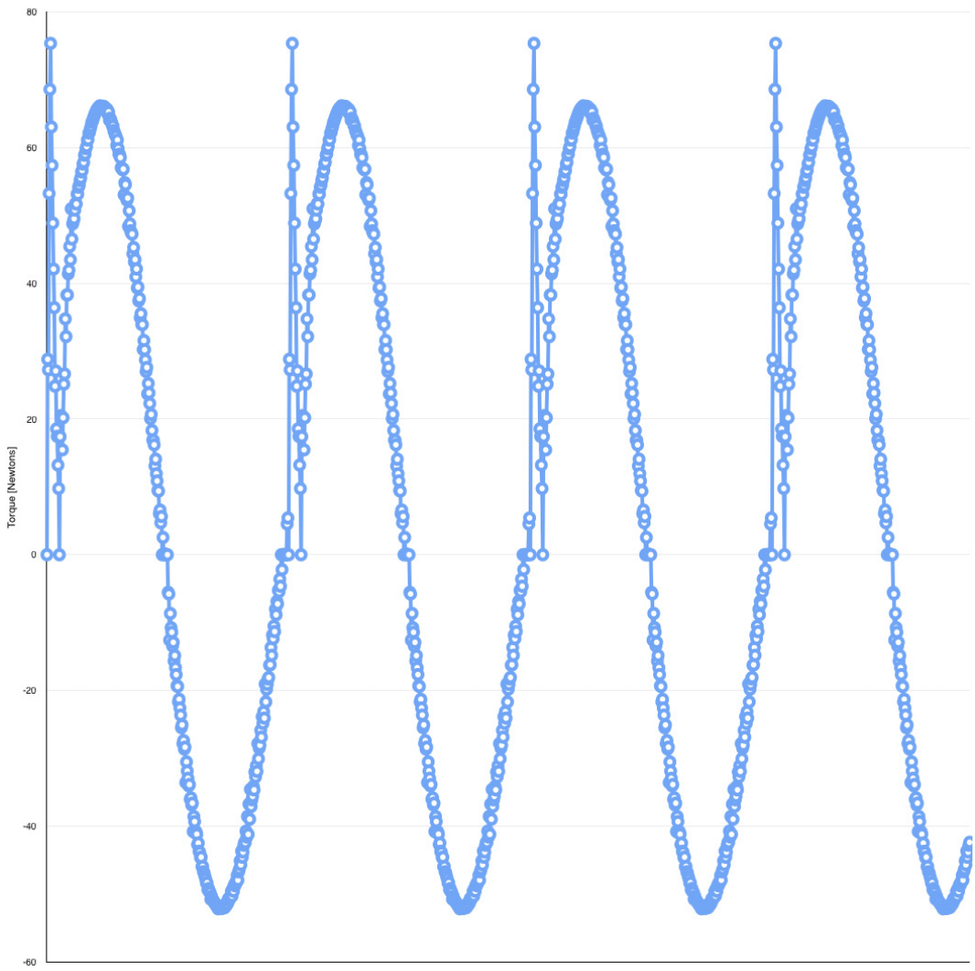


Figure 7.44: Fixed Base Torque, Ki = 450

7.0.2.2 Joint 2

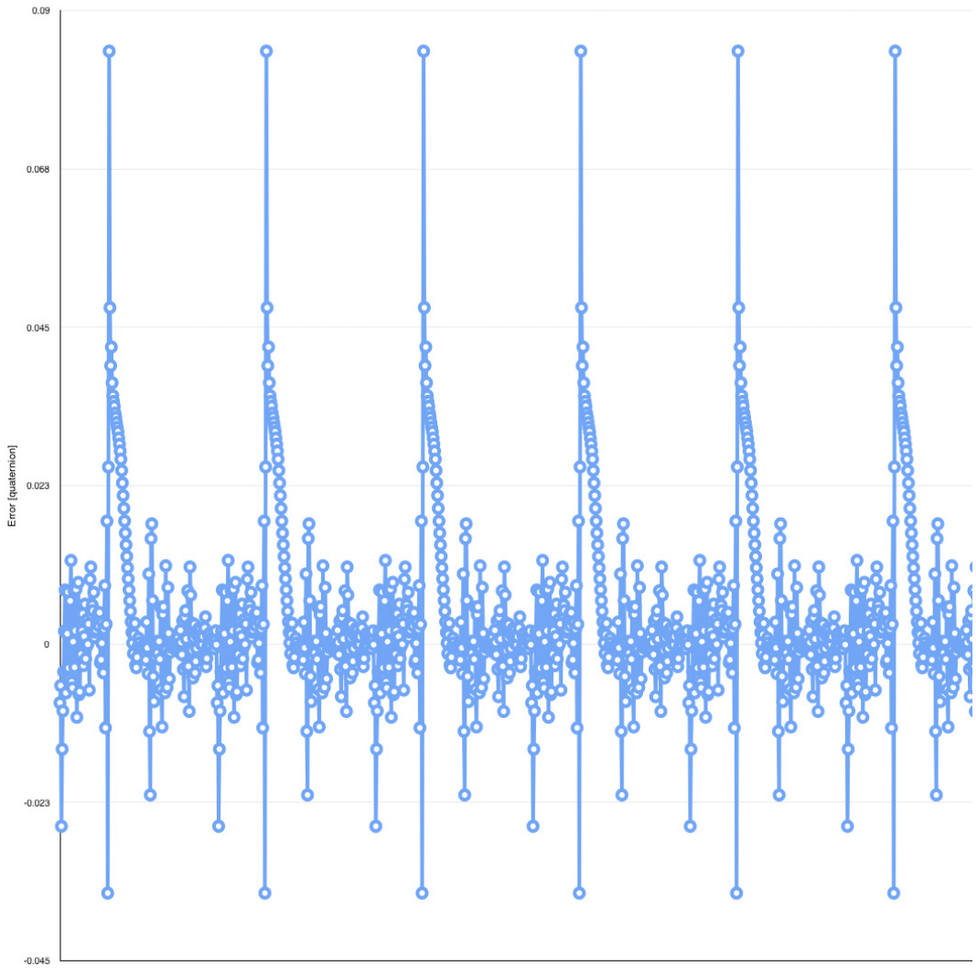


Figure 7.45: Fixed Base Position error, Ki = 100

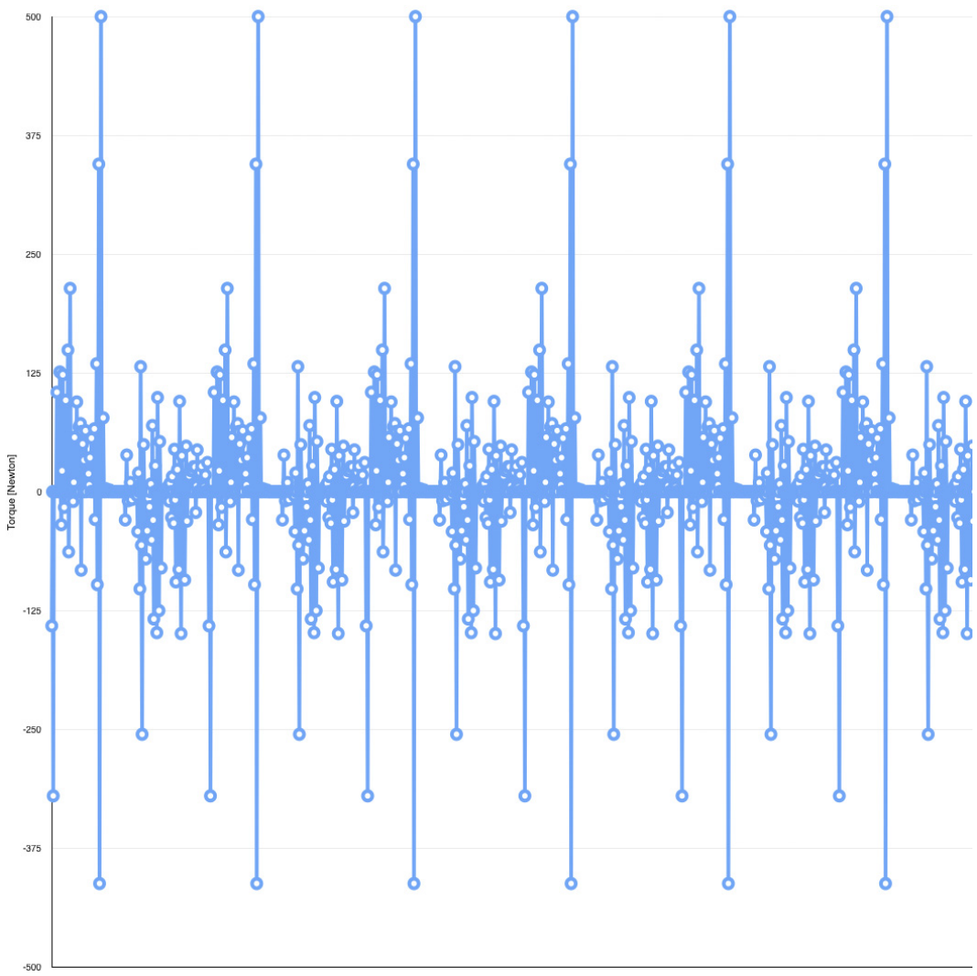


Figure 7.46: Fixed Base Torque, Ki = 100

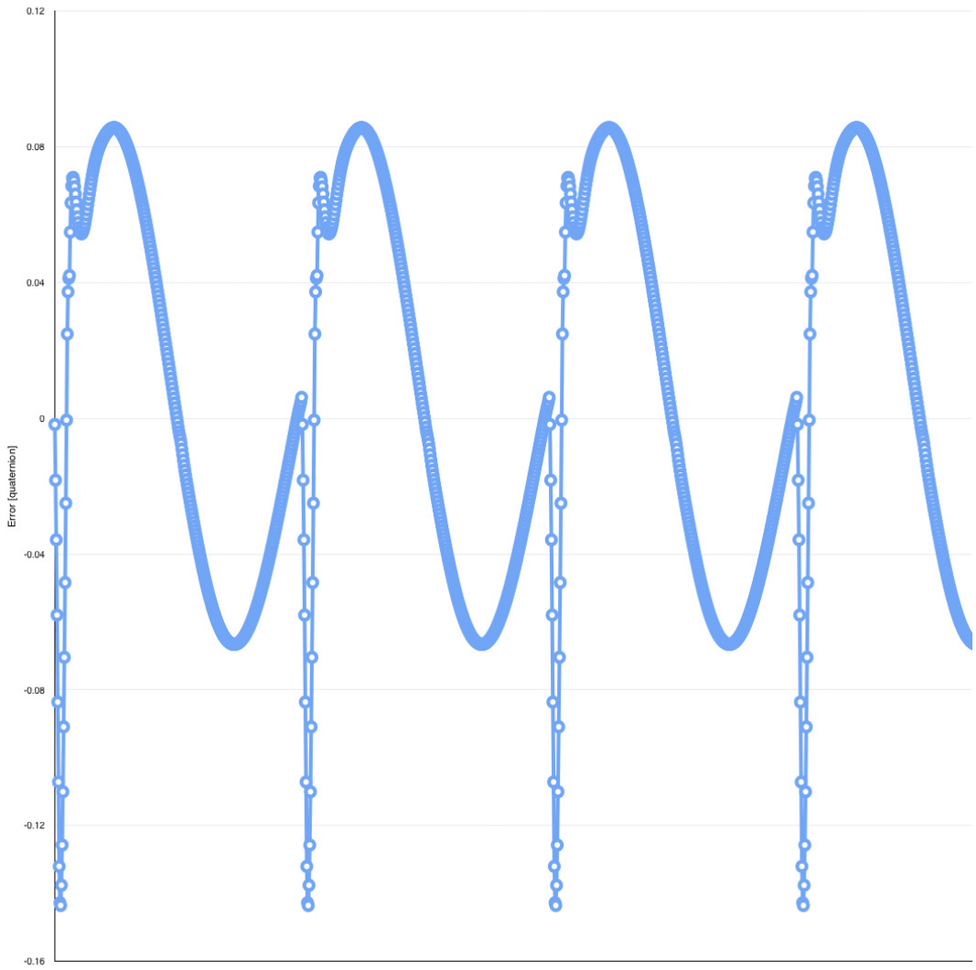


Figure 7.47: Fixed Base Position error, Ki = 150

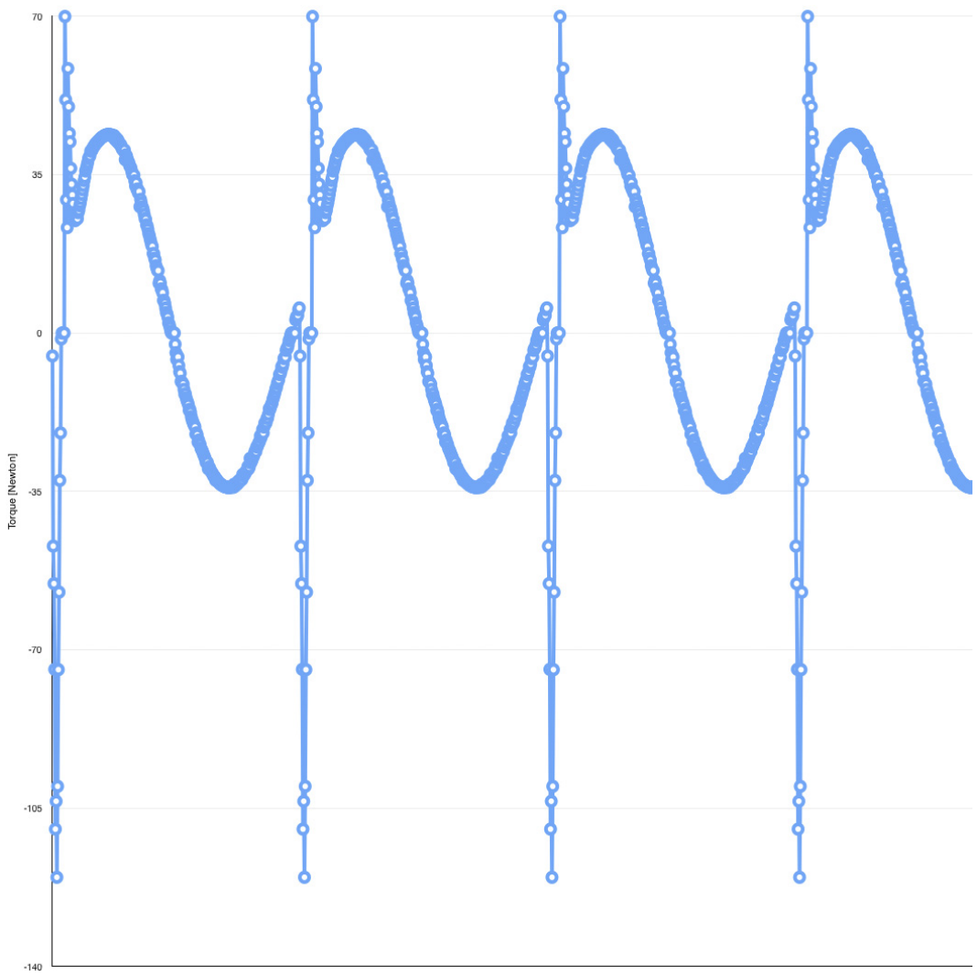


Figure 7.48: Fixed Base Torque, Ki = 150

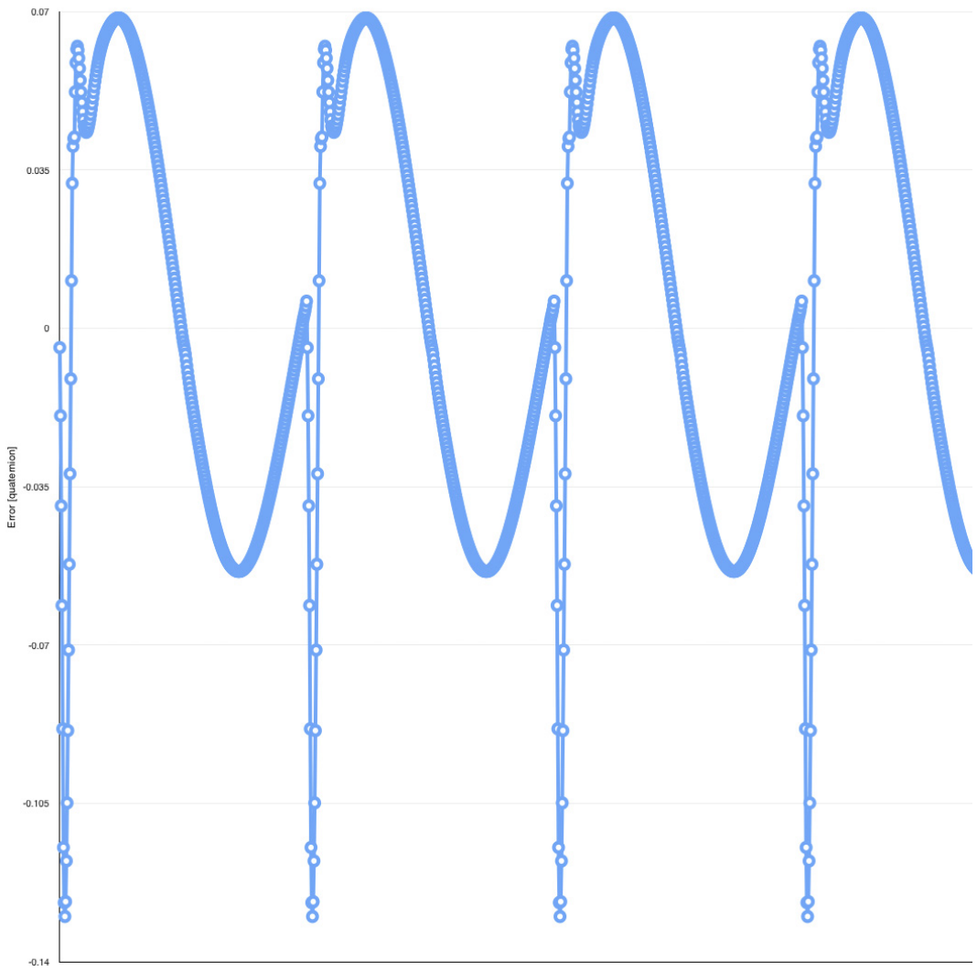


Figure 7.49: Fixed Base Position error, Ki = 200

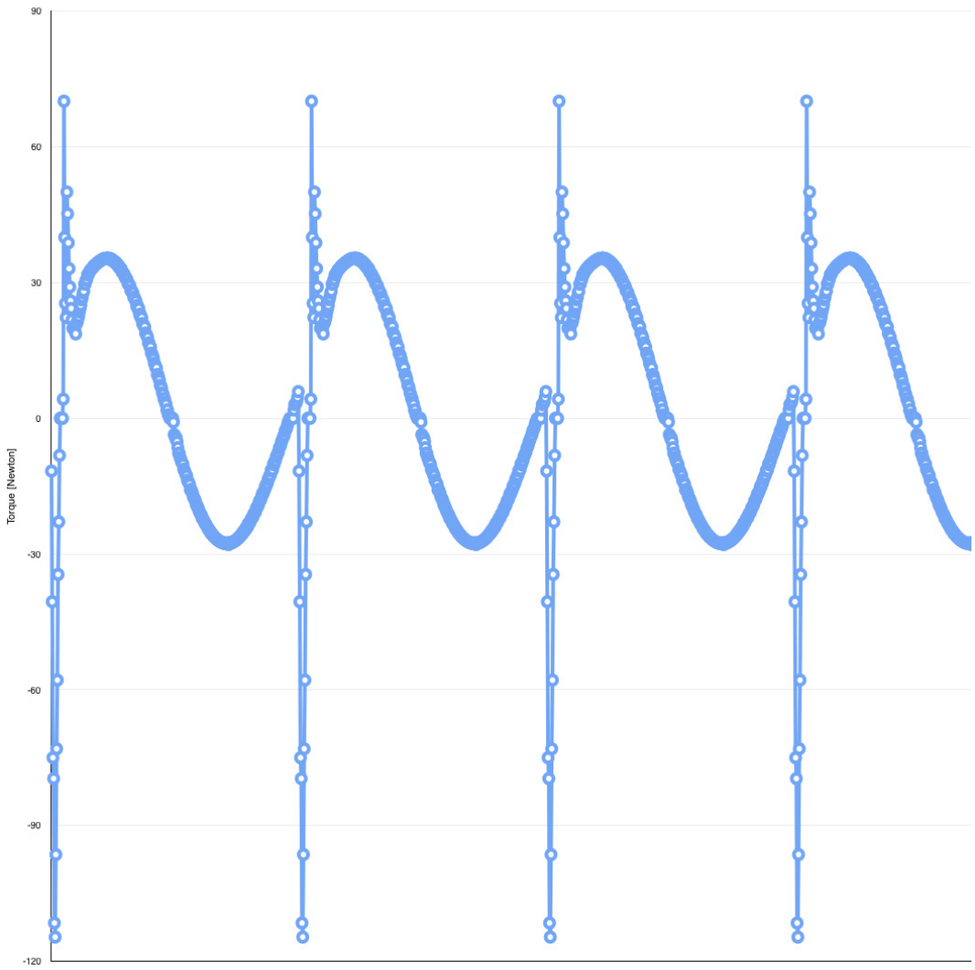


Figure 7.50: Fixed Base Torque, Ki = 200

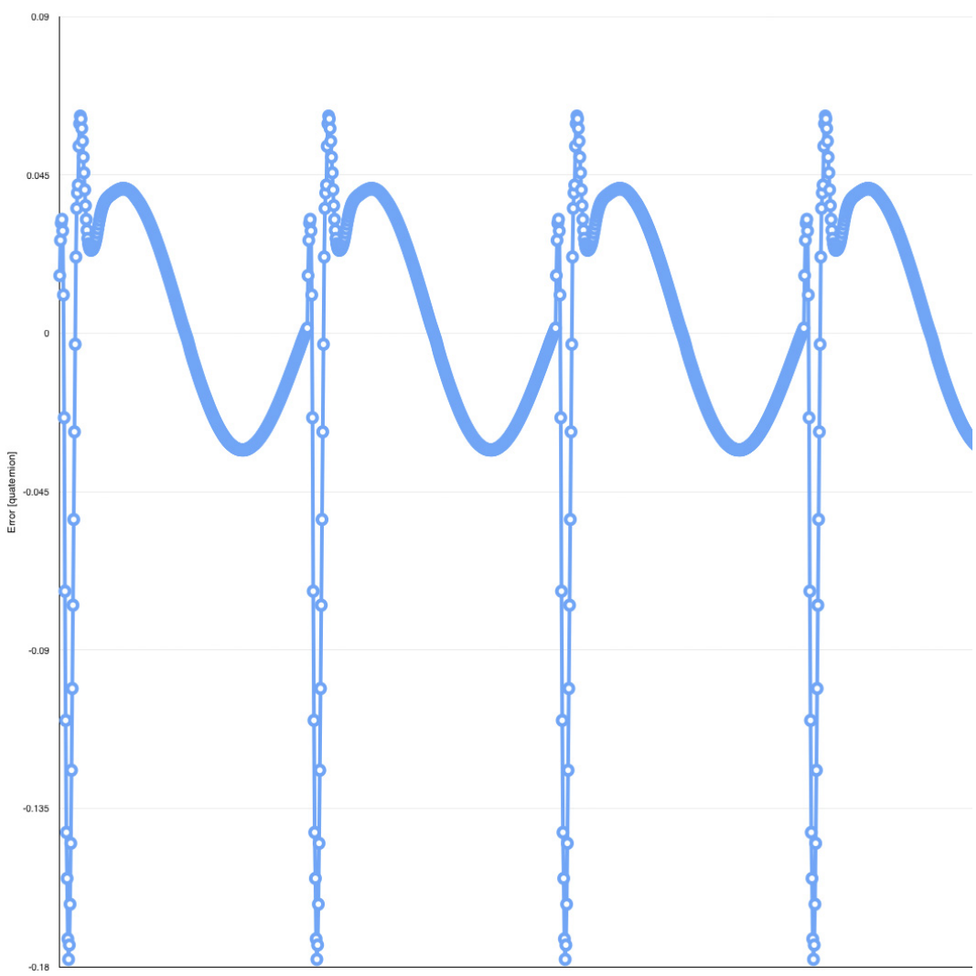


Figure 7.51: Fixed Base Position error, Ki = 250

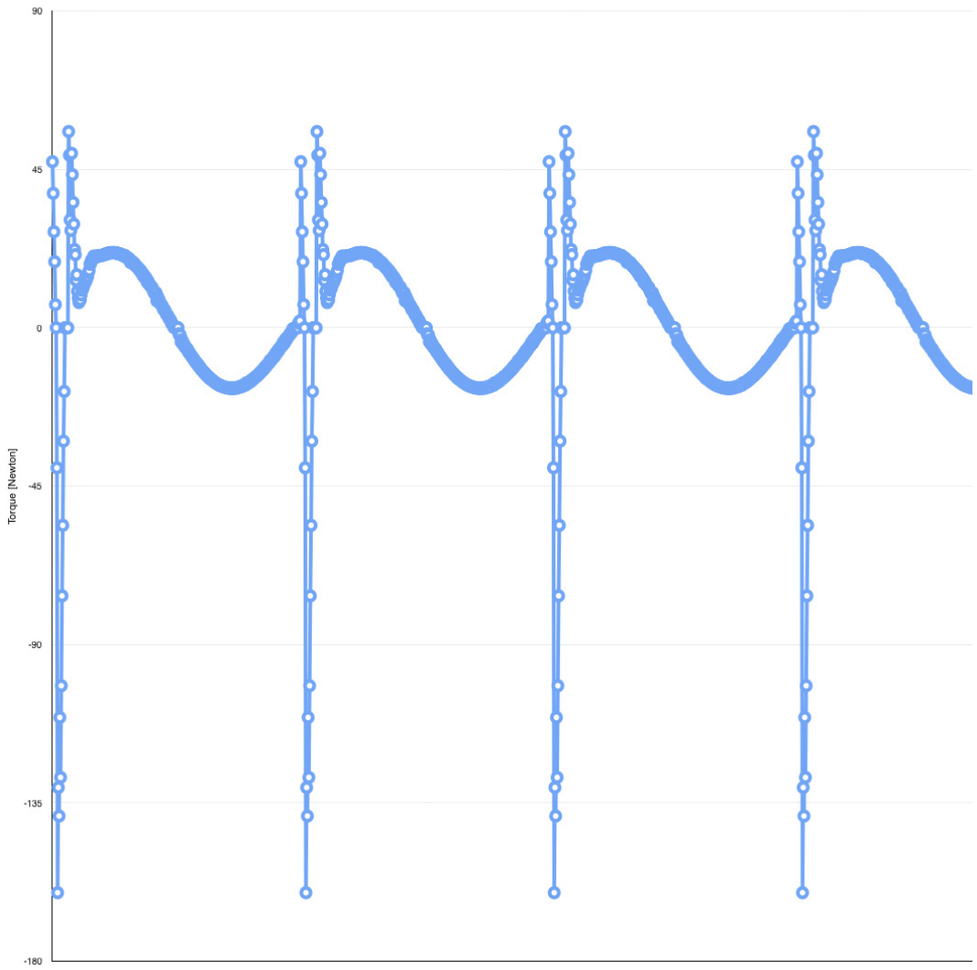


Figure 7.52: Fixed Base Torque, Ki = 250

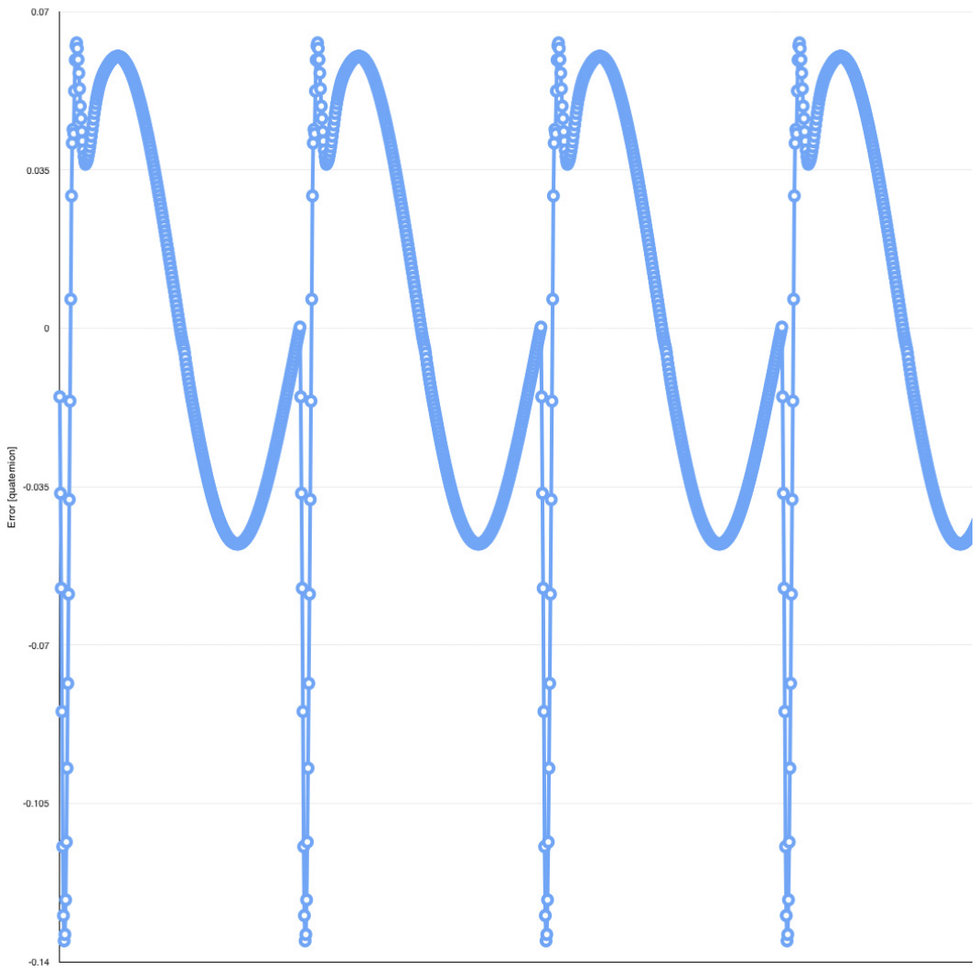


Figure 7.53: Fixed Base Position error, Ki = 300

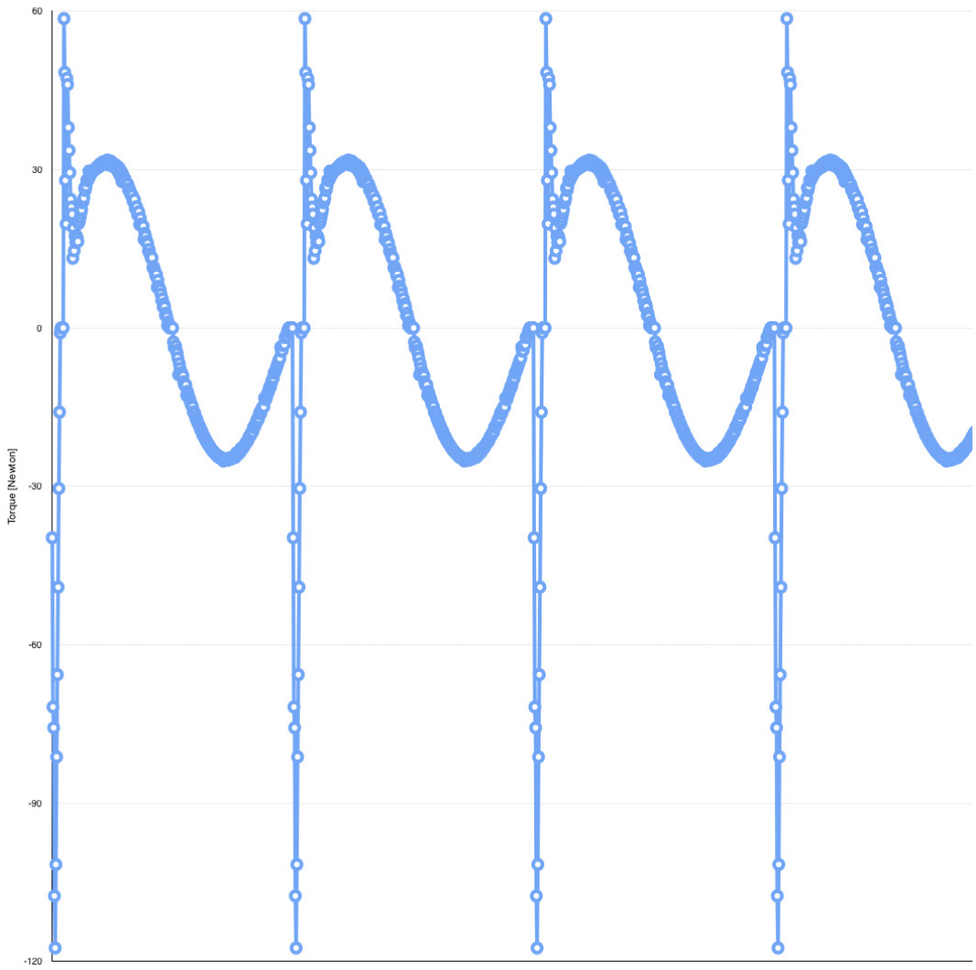


Figure 7.54: Fixed Base Torque, Ki = 300

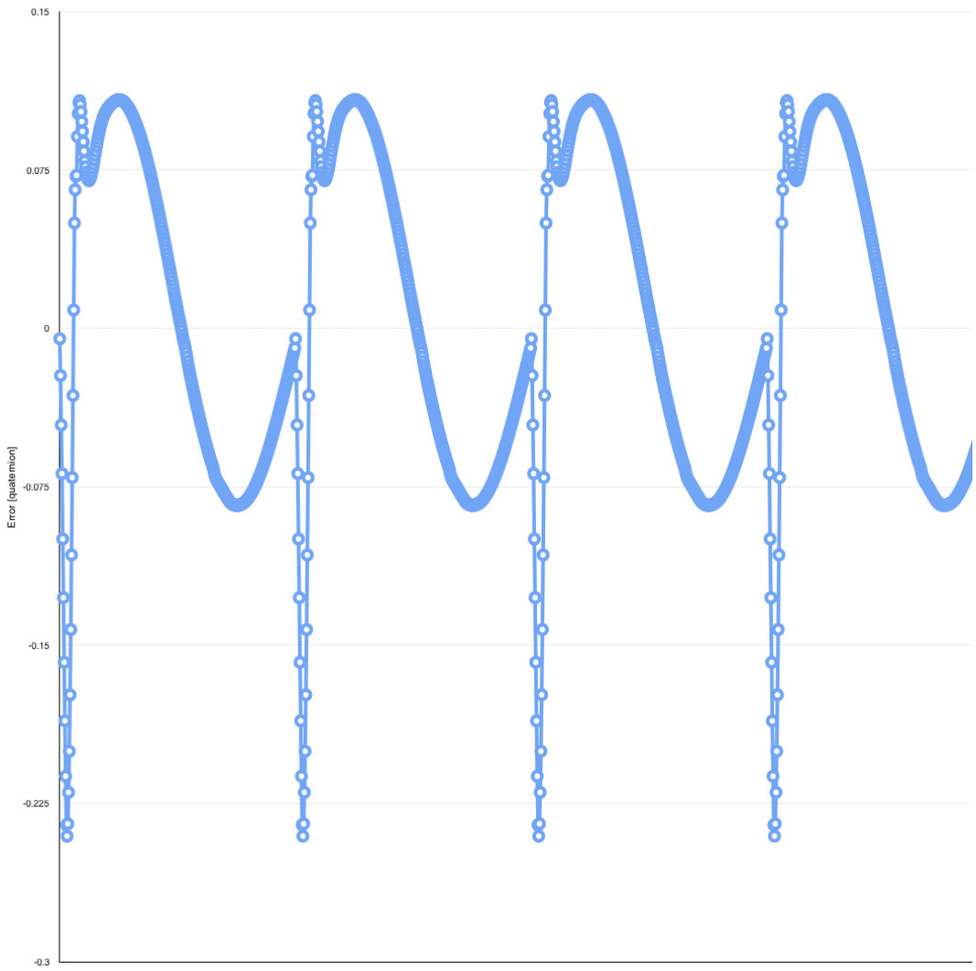


Figure 7.55: Fixed Base Position error, Ki = 350

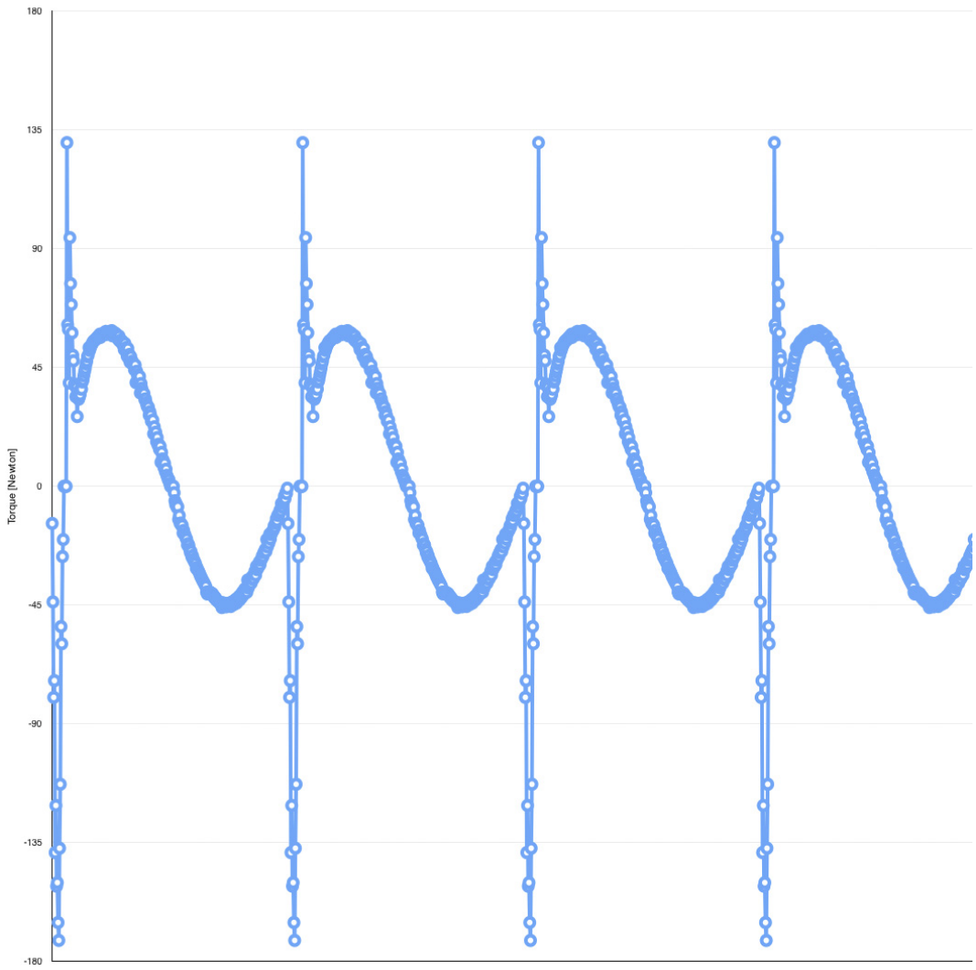


Figure 7.56: Fixed Base Torque, Ki = 350

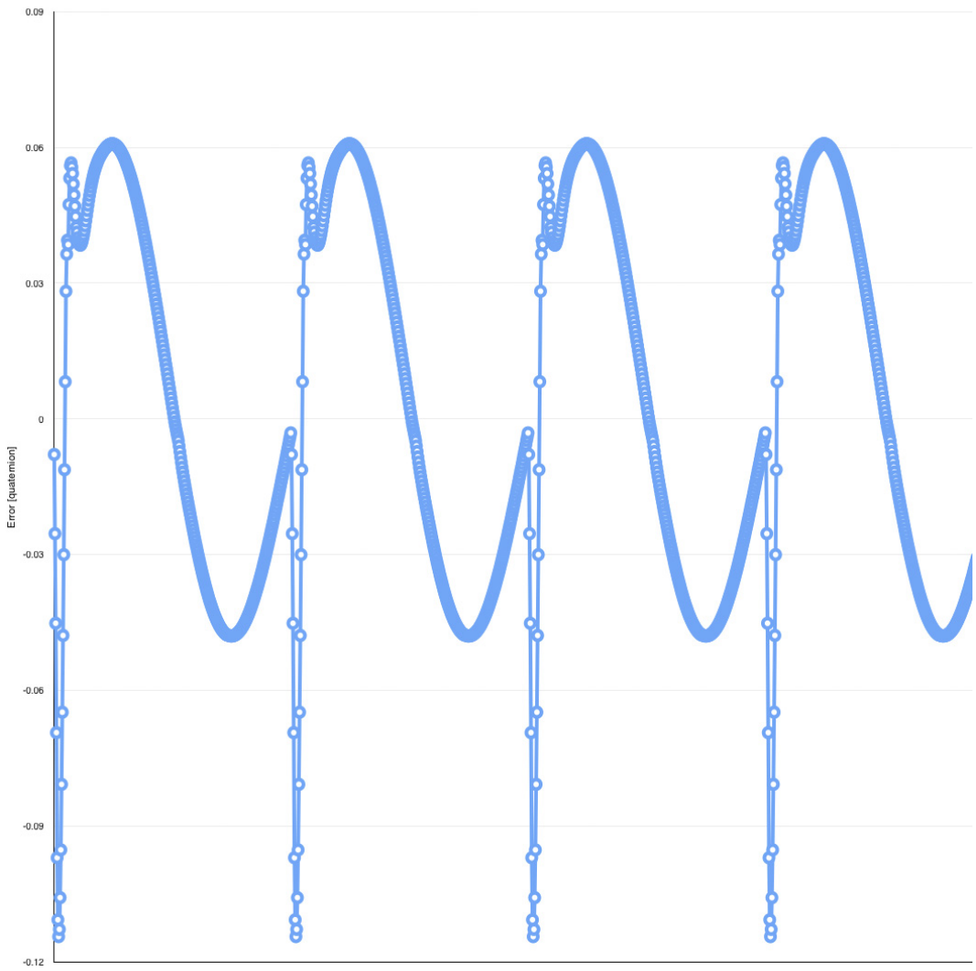


Figure 7.57: Fixed Base Position error, Ki = 400

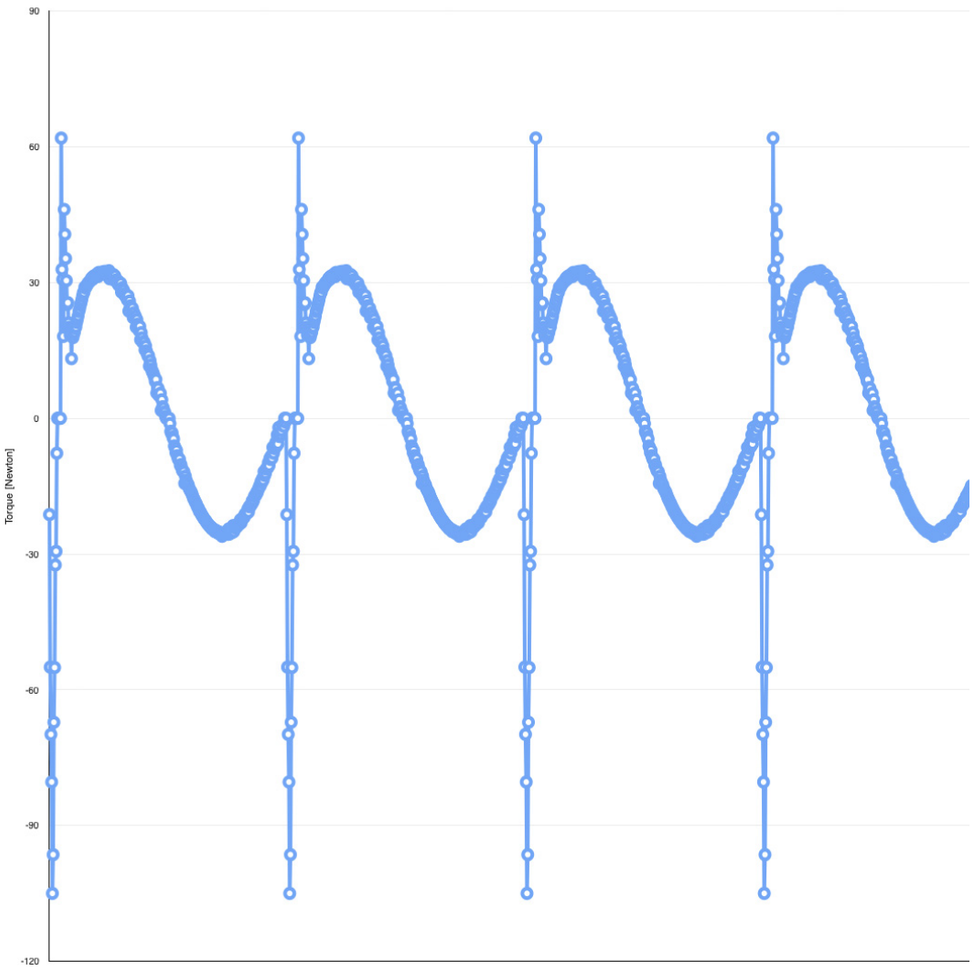


Figure 7.58: Fixed Base Torque, Ki = 400

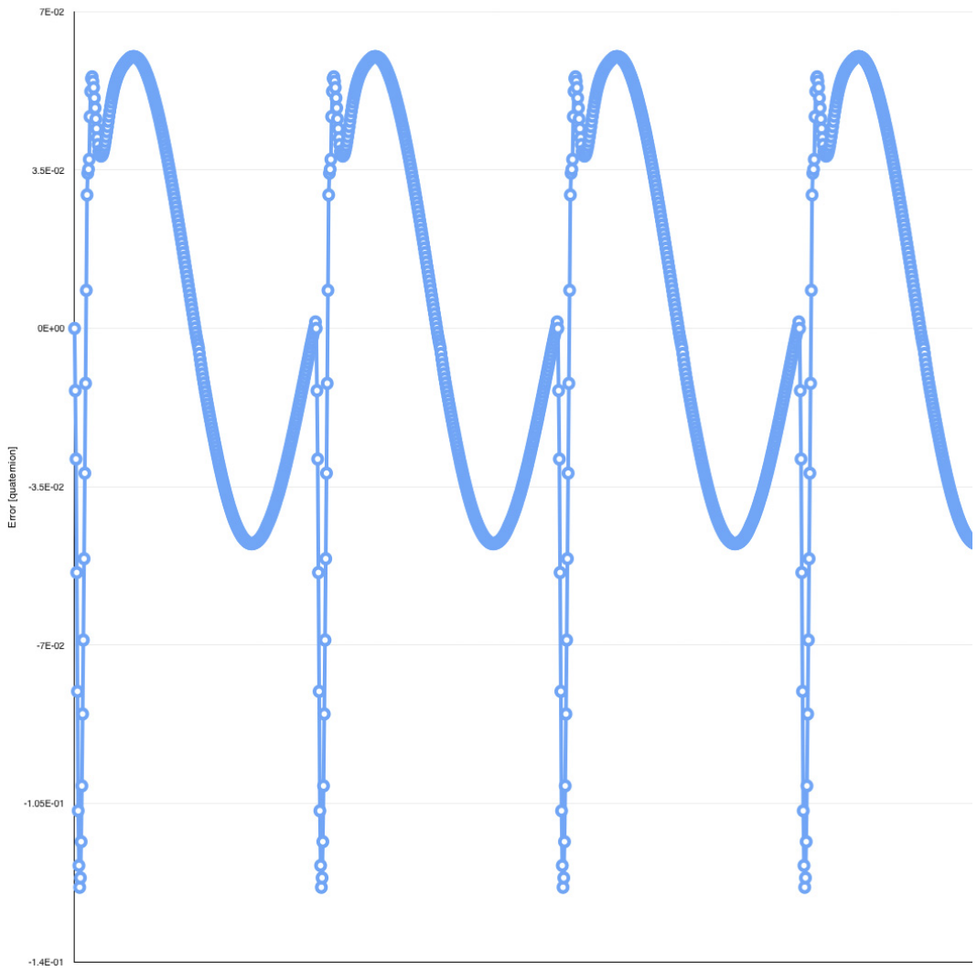


Figure 7.59: Fixed Base Position error, Ki = 450

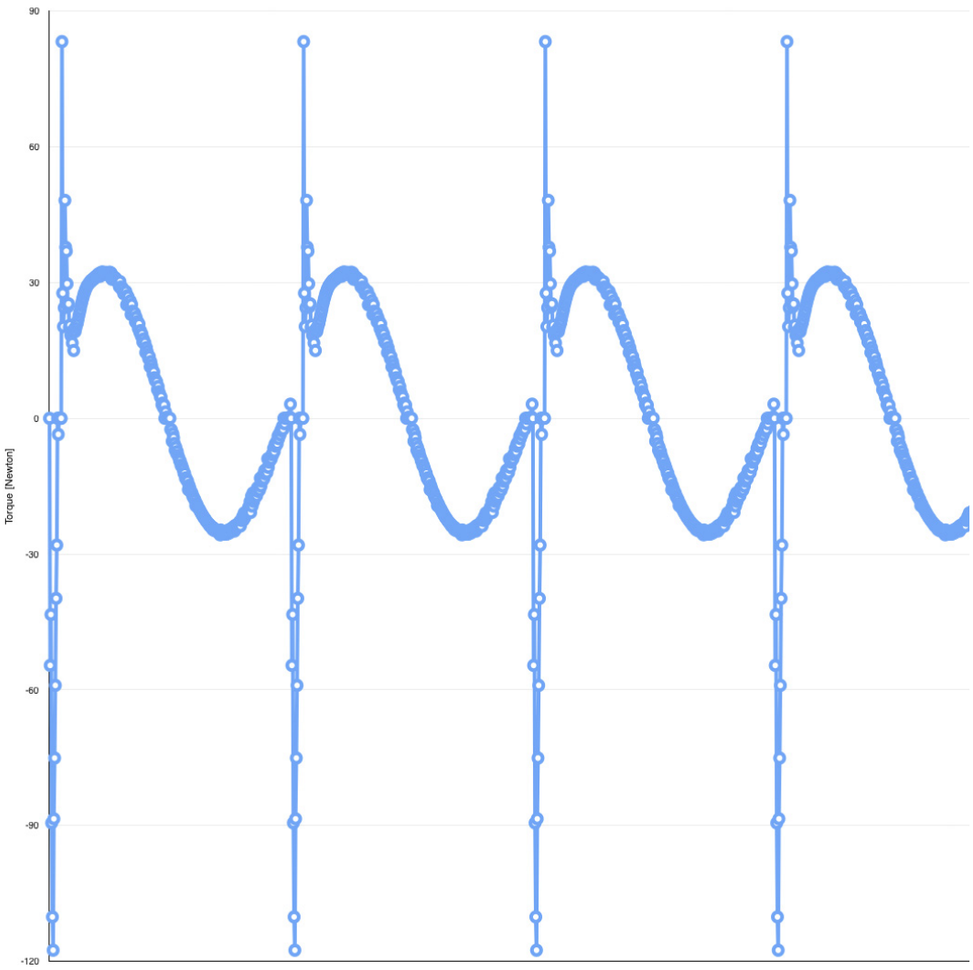


Figure 7.60: Fixed Base Torque, Ki = 450

7.0.3 Derivative variable

7.0.3.1 Joint 1

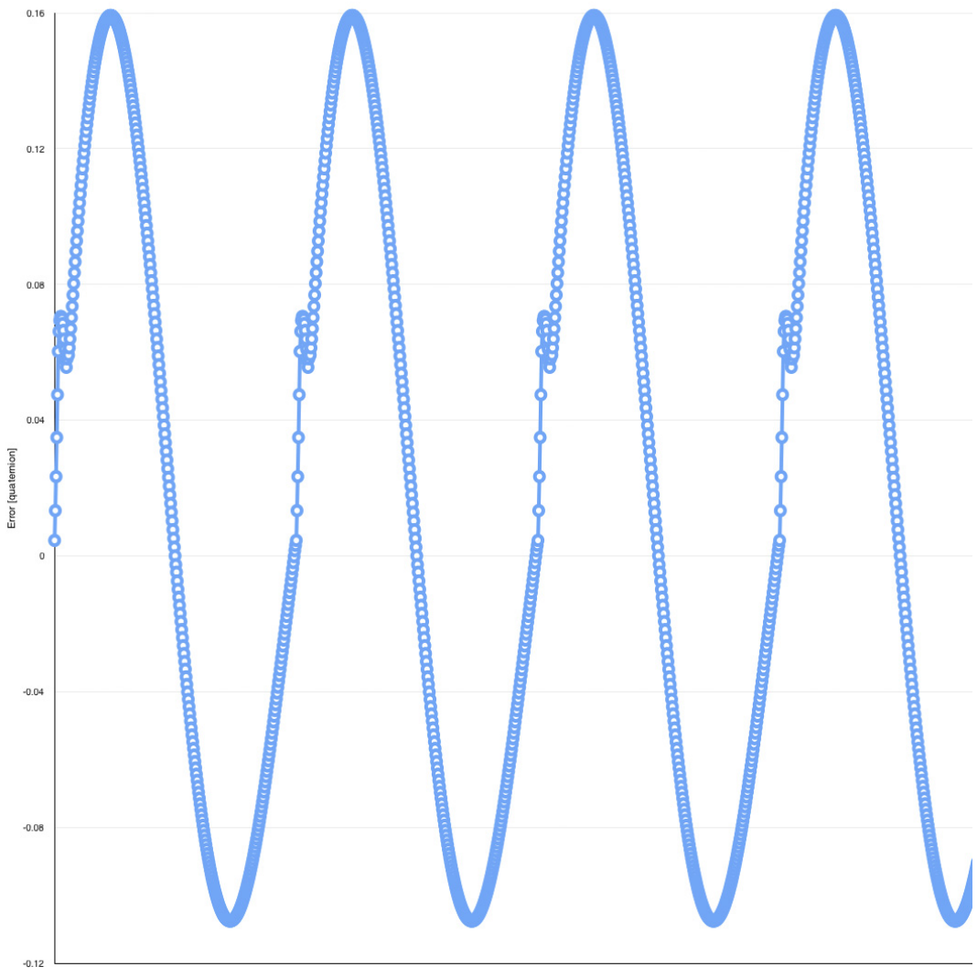


Figure 7.61: Fixed Base Position error, Kd = 100

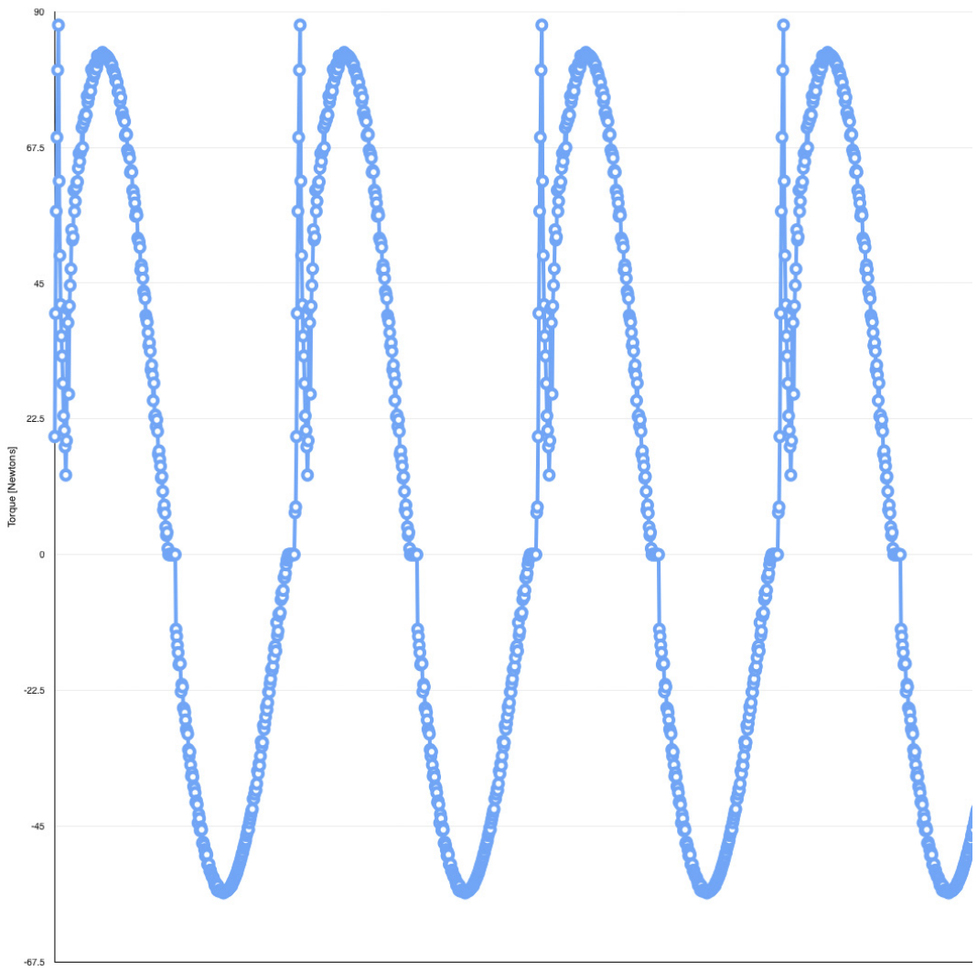


Figure 7.62: Fixed Base Torque, Kd = 100

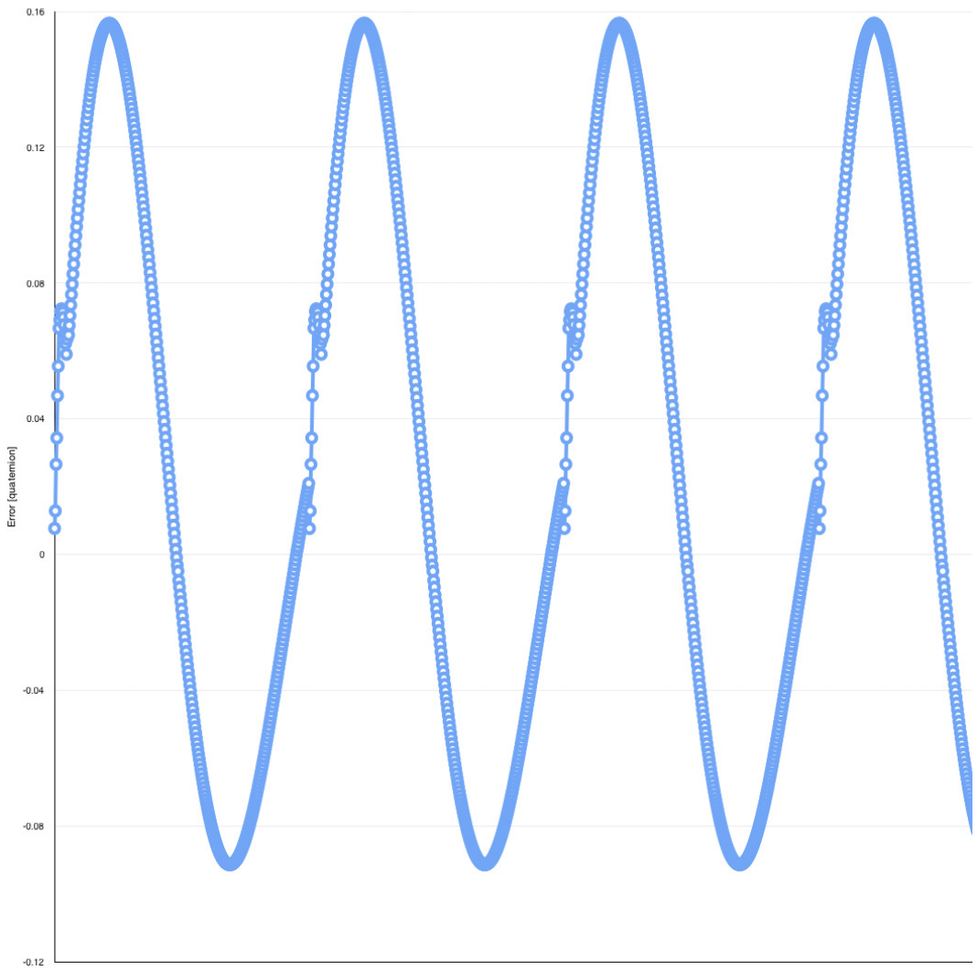


Figure 7.63: Fixed Base Position error, Kd = 150

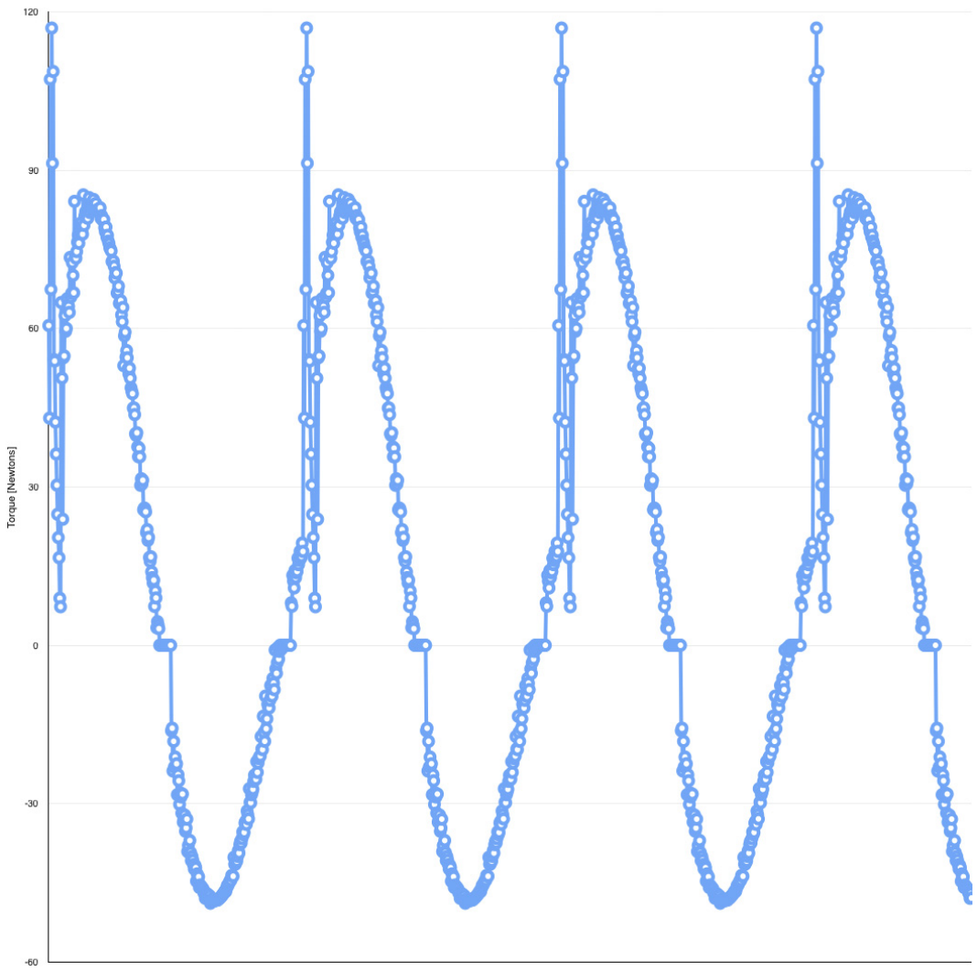


Figure 7.64: Fixed Base Torque, Kd = 150

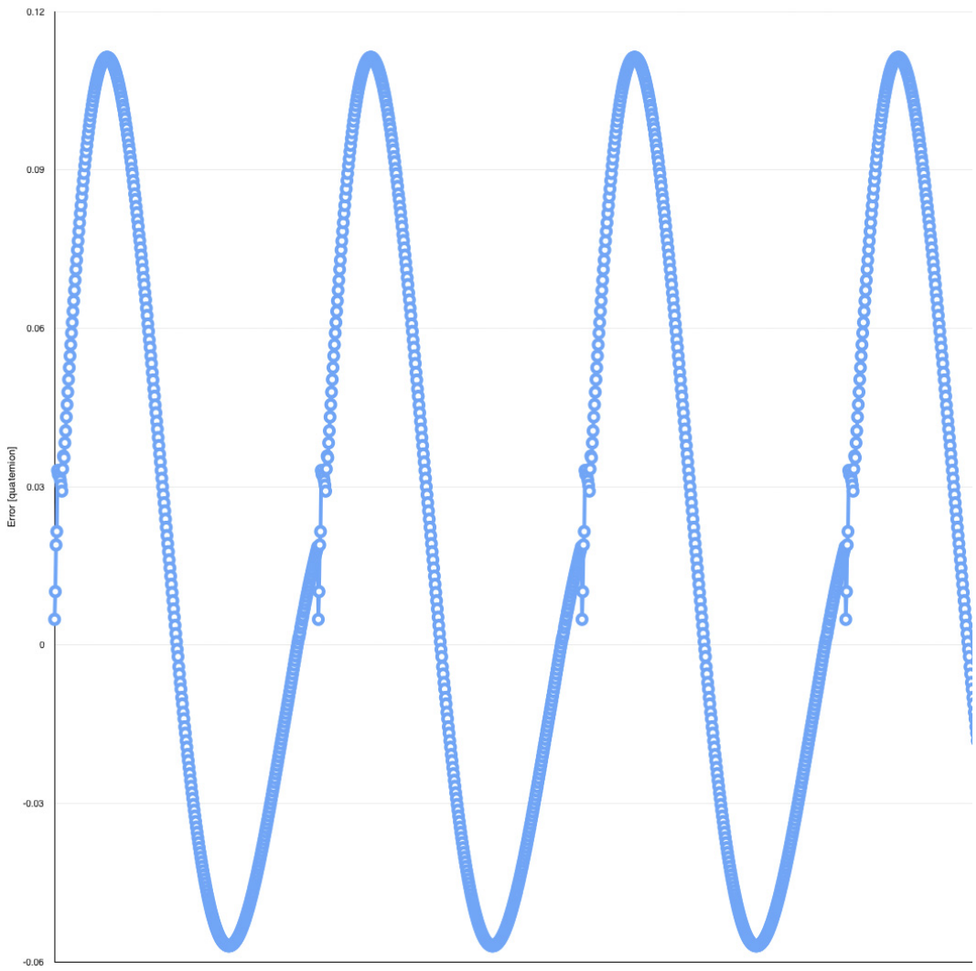


Figure 7.65: Fixed Base Position error, Kd = 200

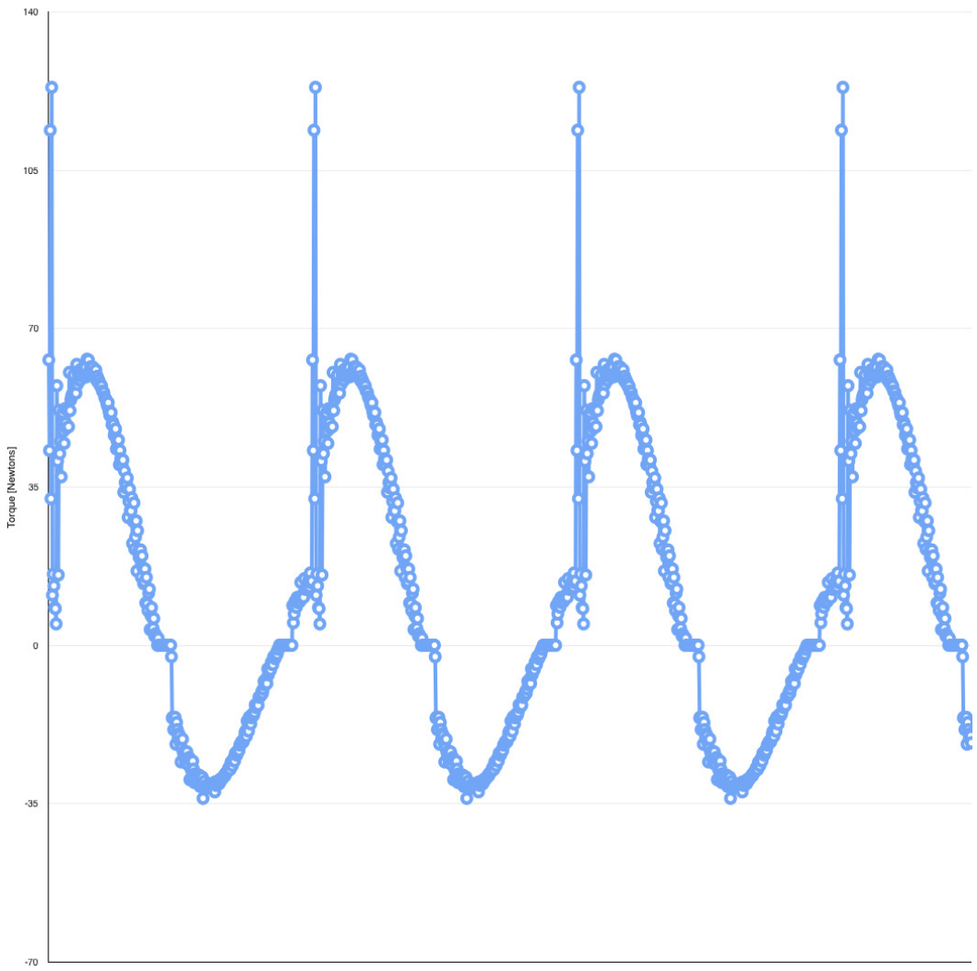


Figure 7.66: Fixed Base Torque, Kd = 200

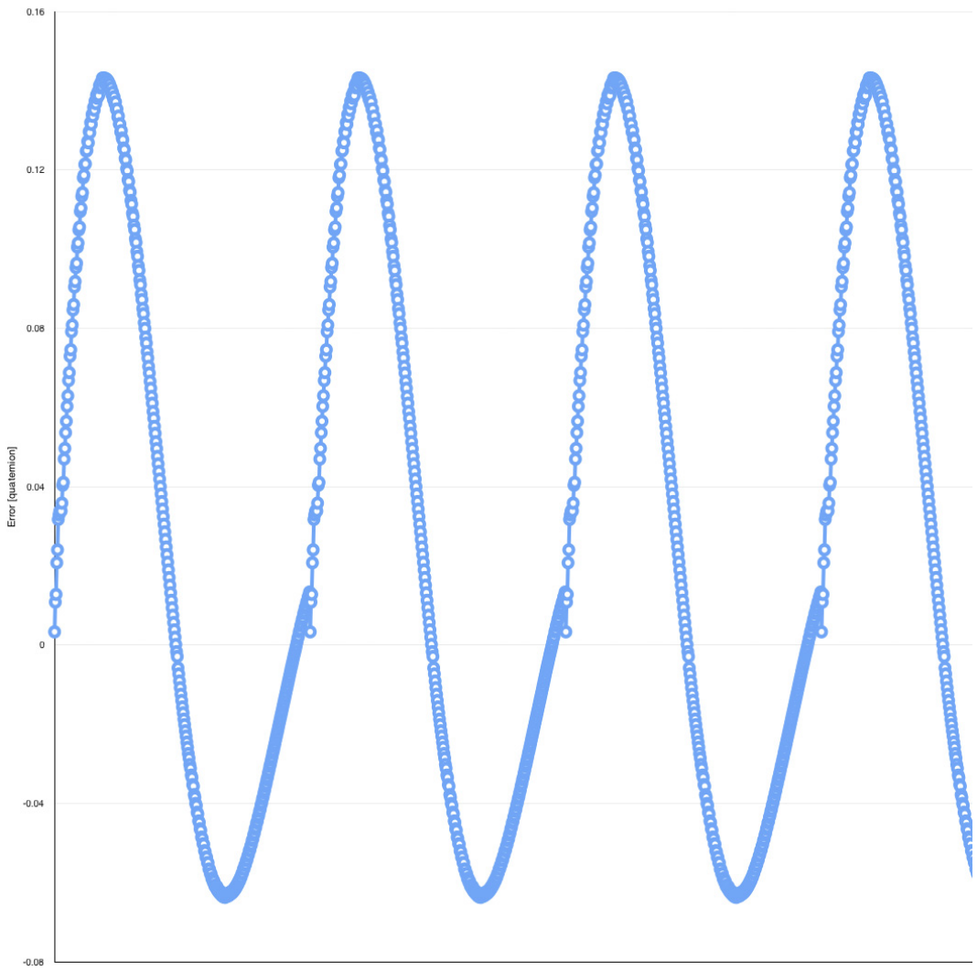


Figure 7.67: Fixed Base Position error, Kd = 250

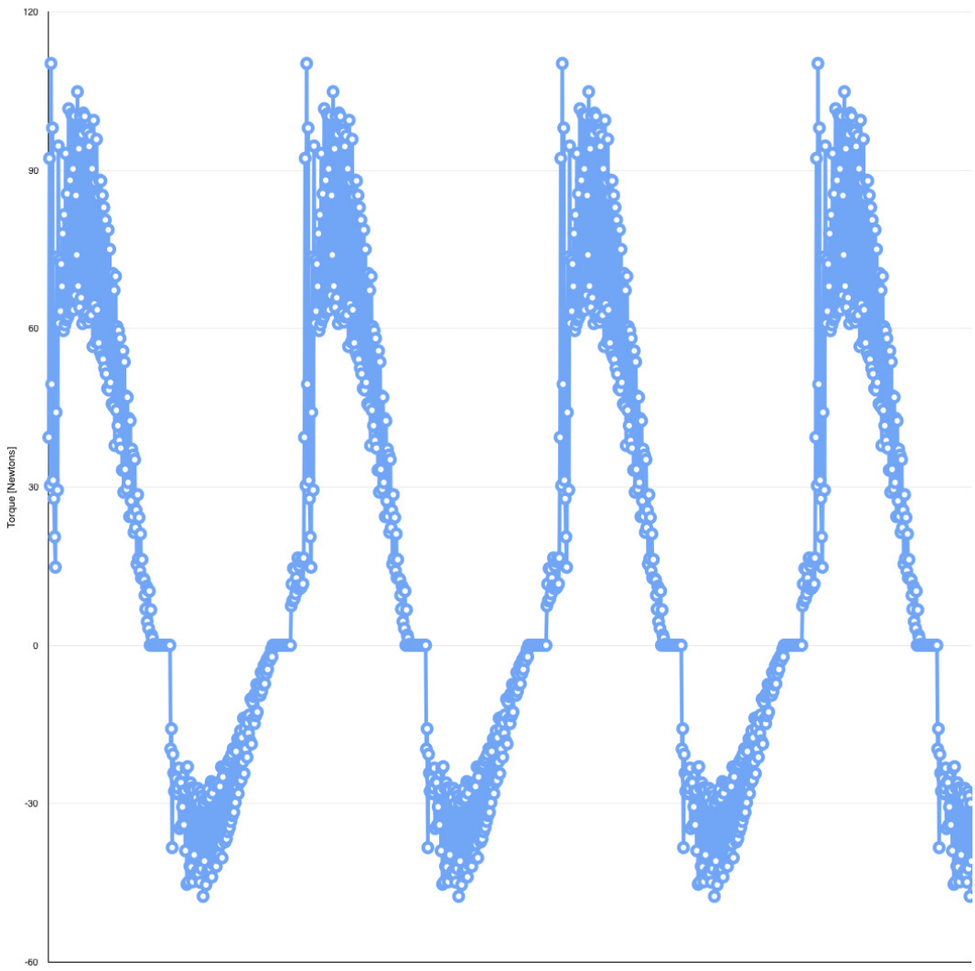


Figure 7.68: Fixed Base Torque, Kd = 250

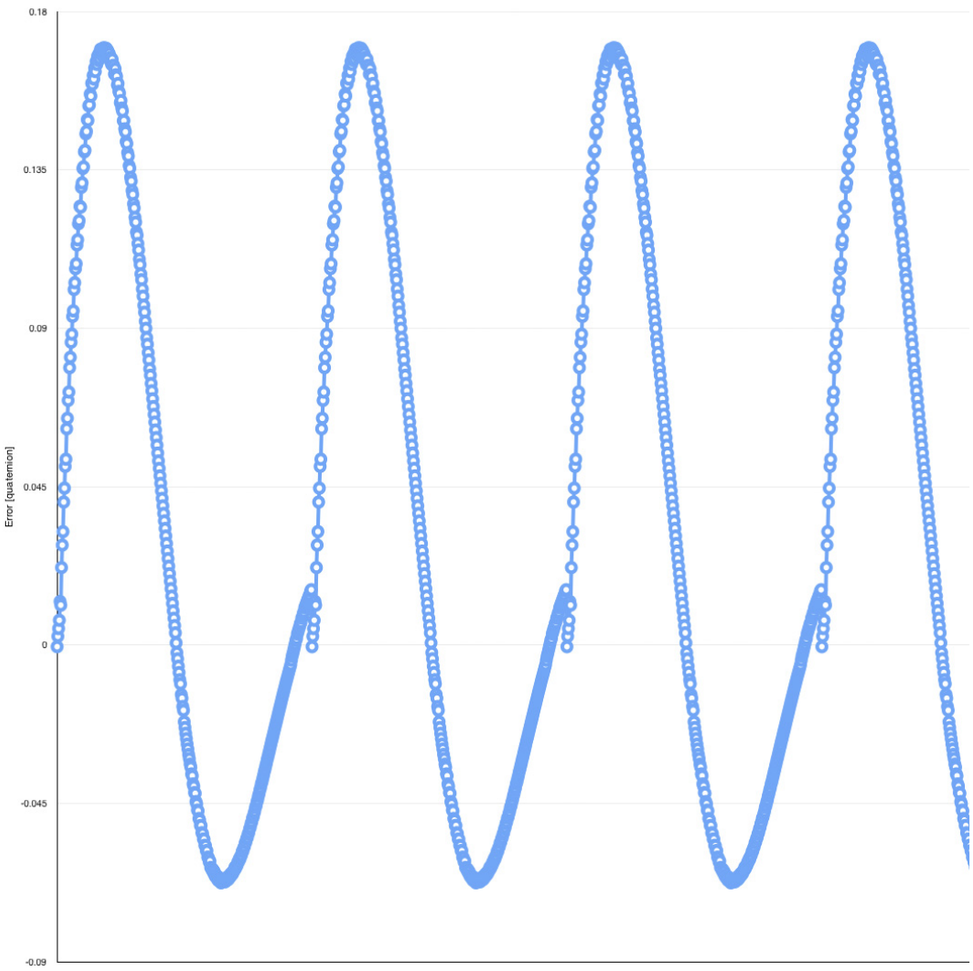


Figure 7.69: Fixed Base Position error, Kd = 300

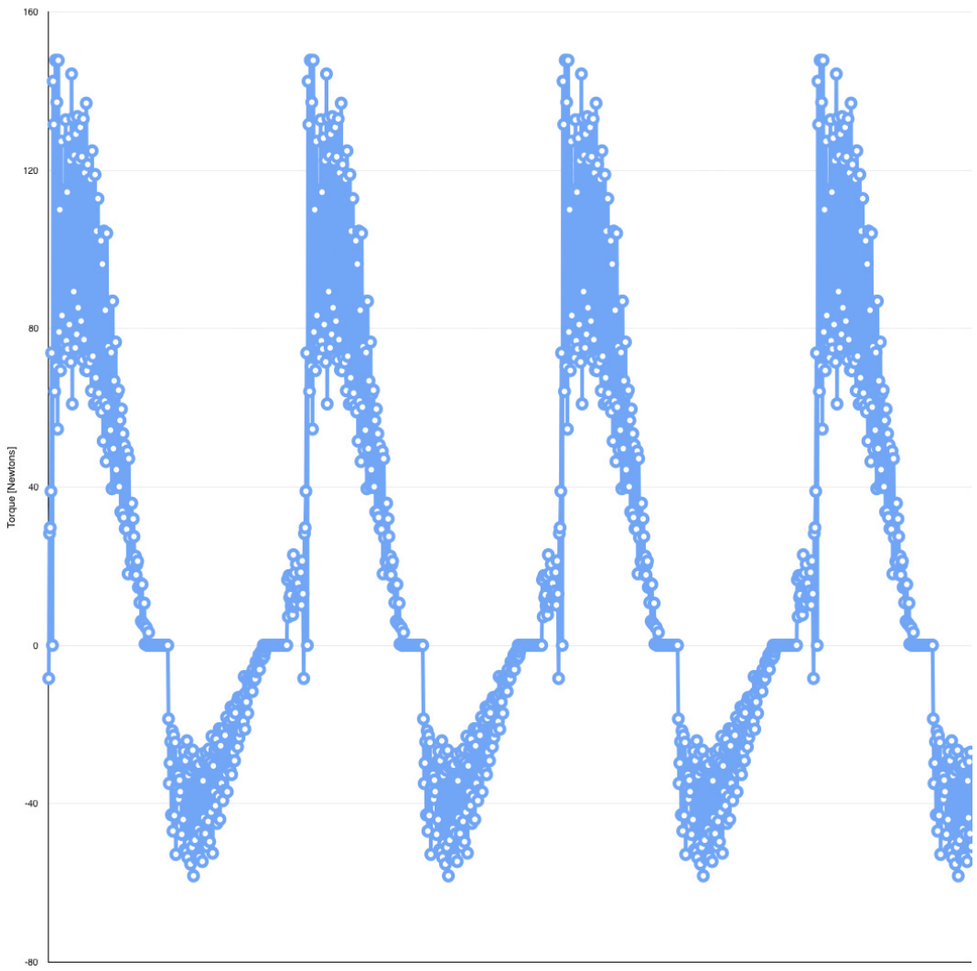


Figure 7.70: Fixed Base Torque, Kd = 300

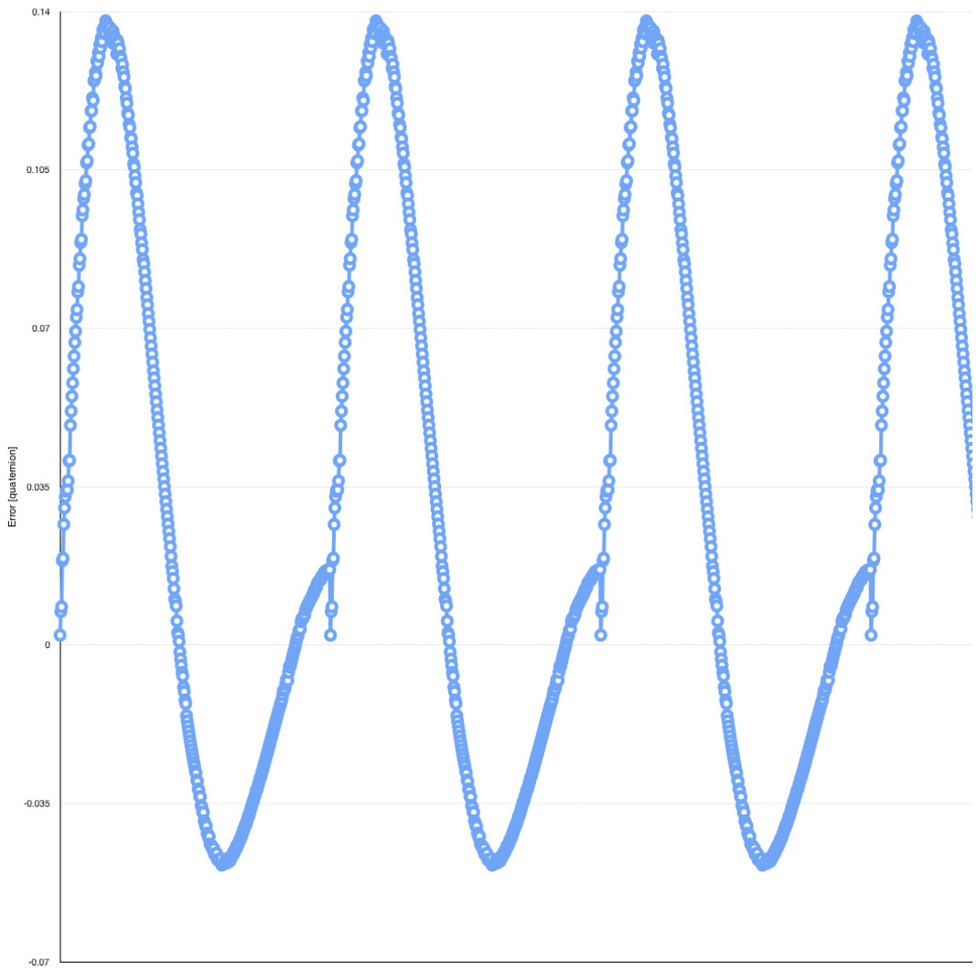


Figure 7.71: Fixed Base Position error, Kd = 350

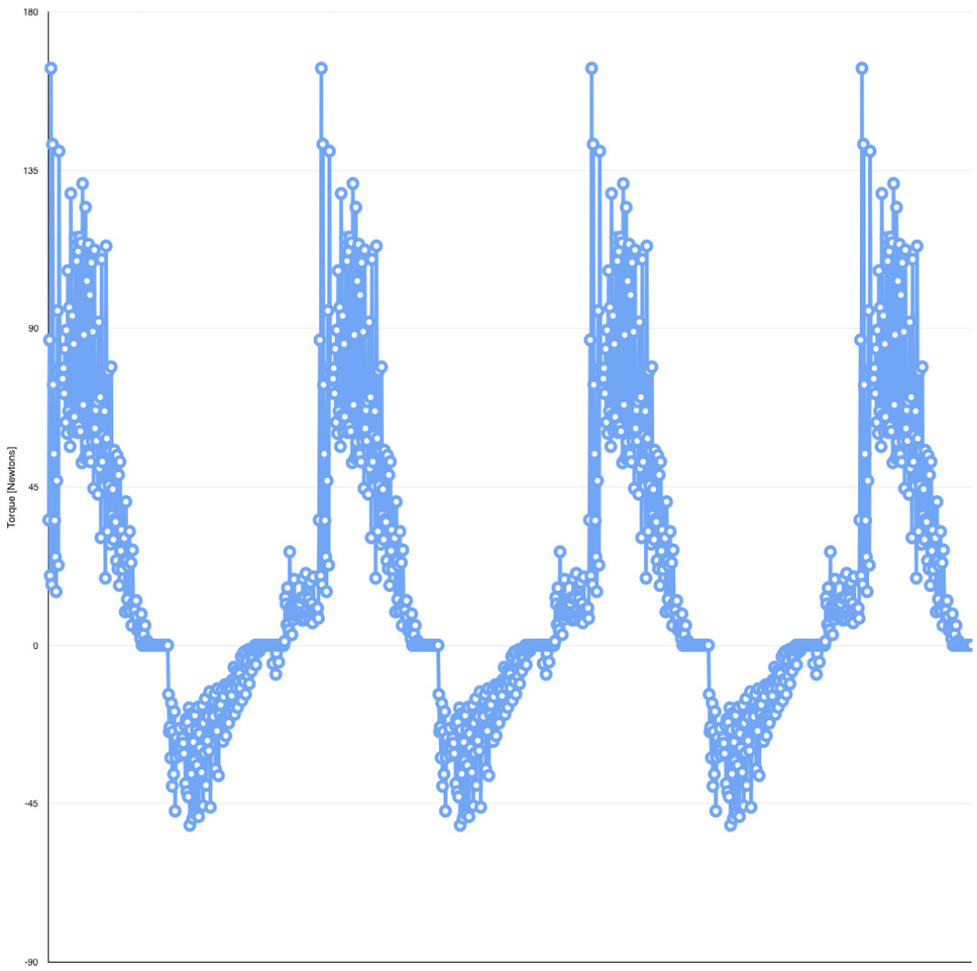


Figure 7.72: Fixed Base Torque, Kd = 350

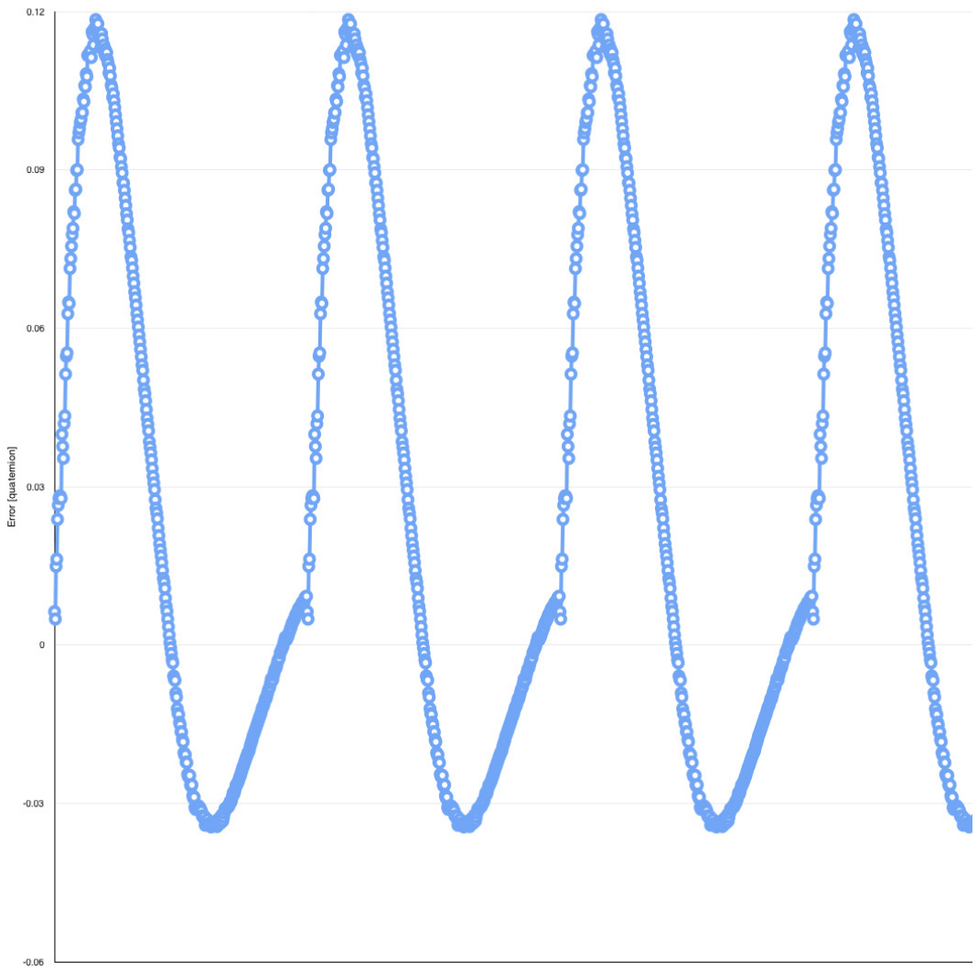


Figure 7.73: Fixed Base Position error, Kd = 400

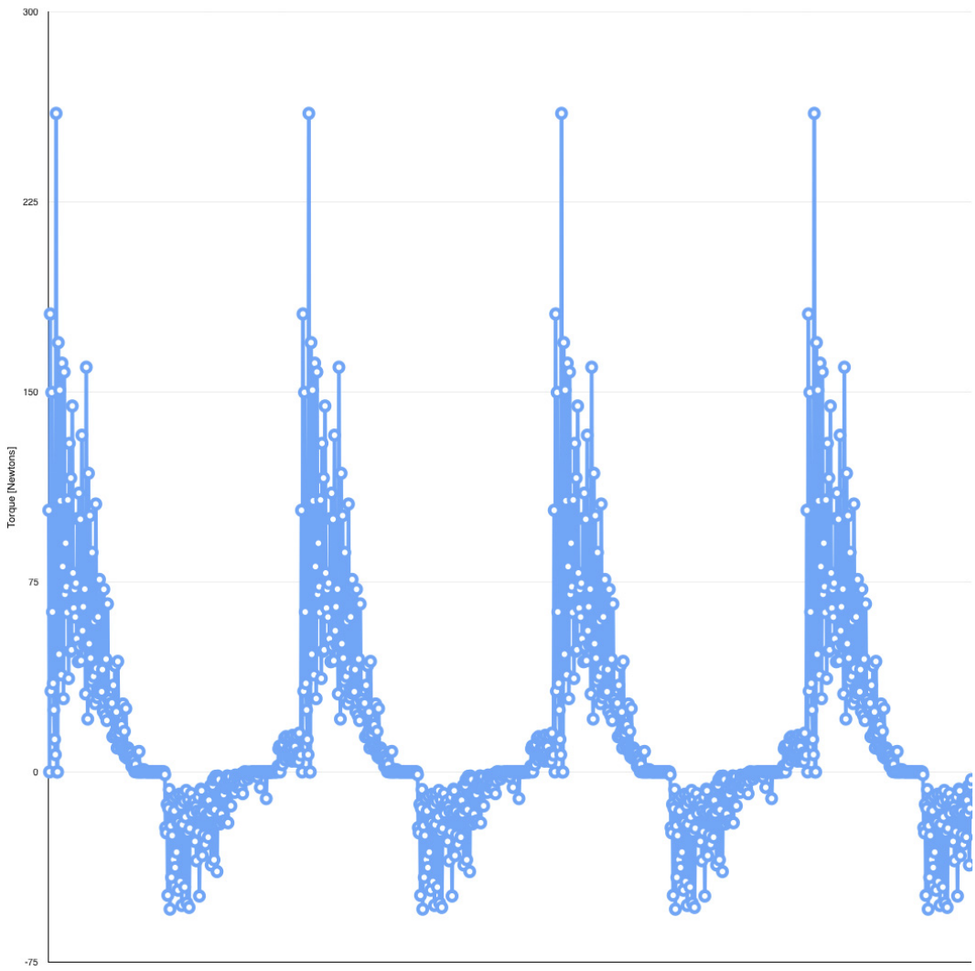


Figure 7.74: Fixed Base Torque, Kd = 400

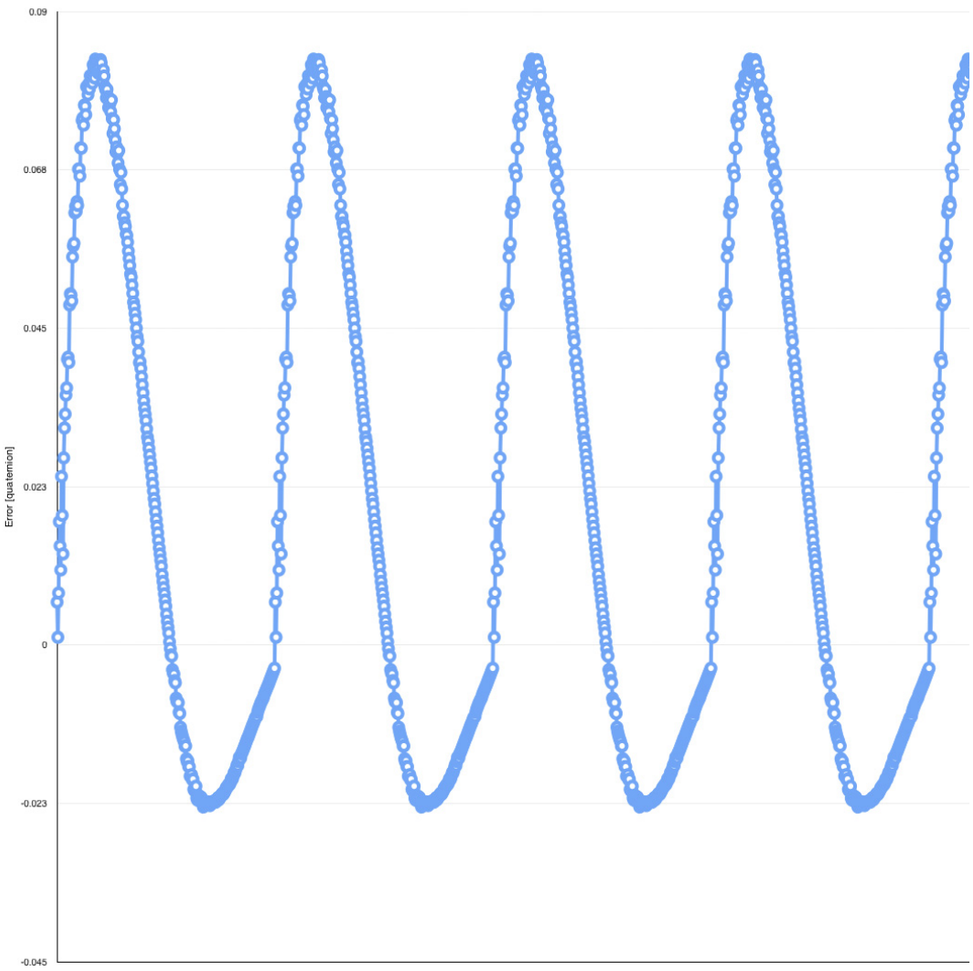


Figure 7.75: Fixed Base Position error, Kd = 450

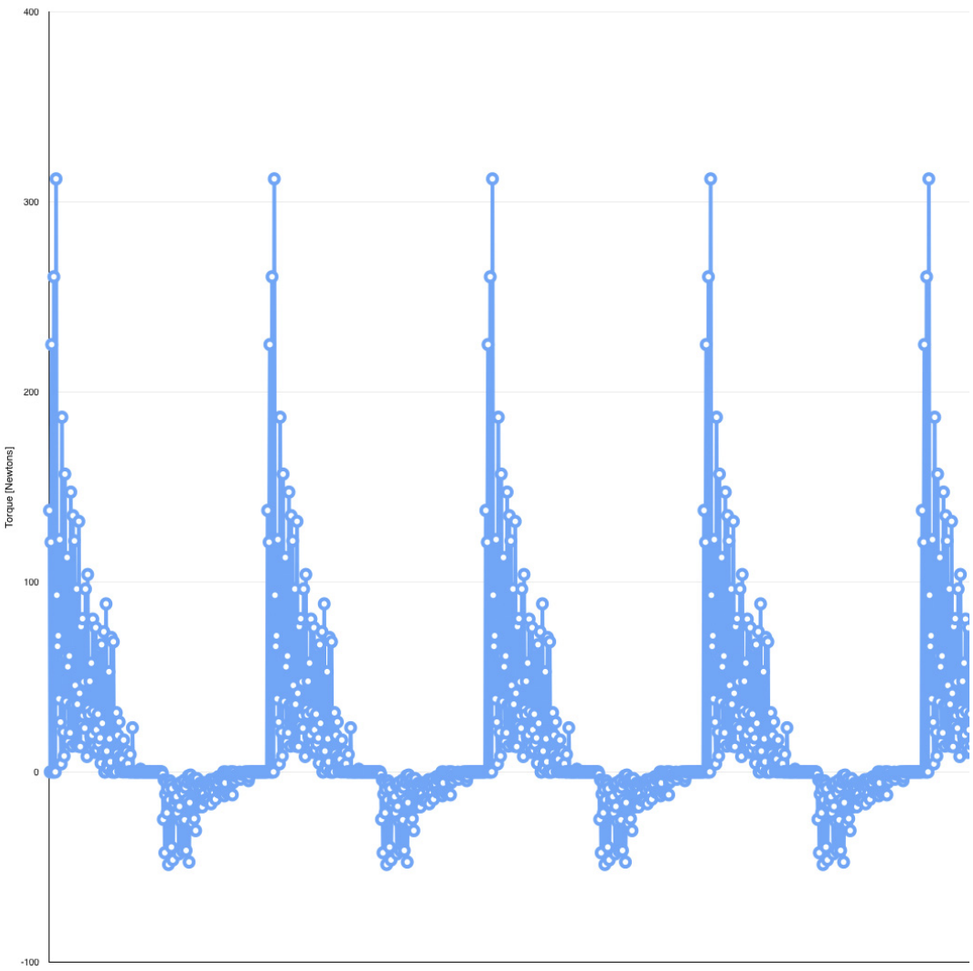


Figure 7.76: Fixed Base Torque, Kd = 450

7.0.3.2 Joint 2

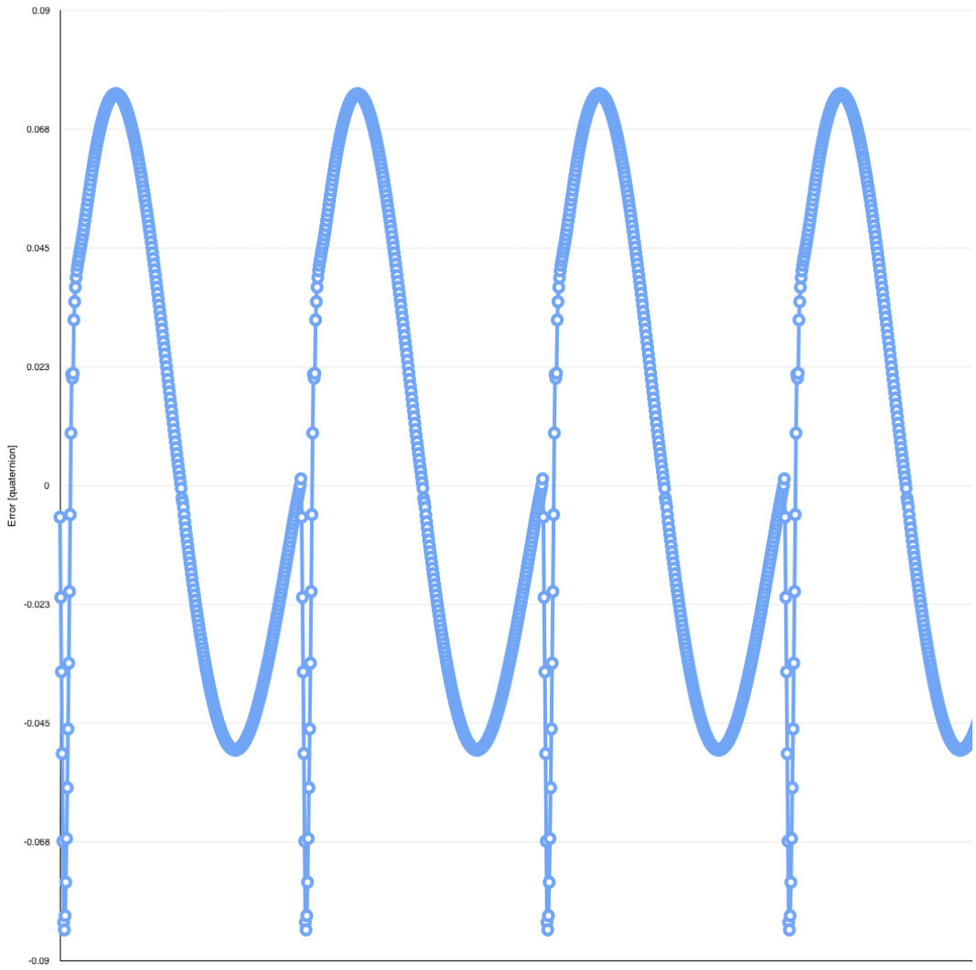


Figure 7.77: Fixed Base Position error, Kd = 100

7.1 Uniform base movement parameter analysis

7.1.1 Proportional variable

7.1.1.1 Joint 1

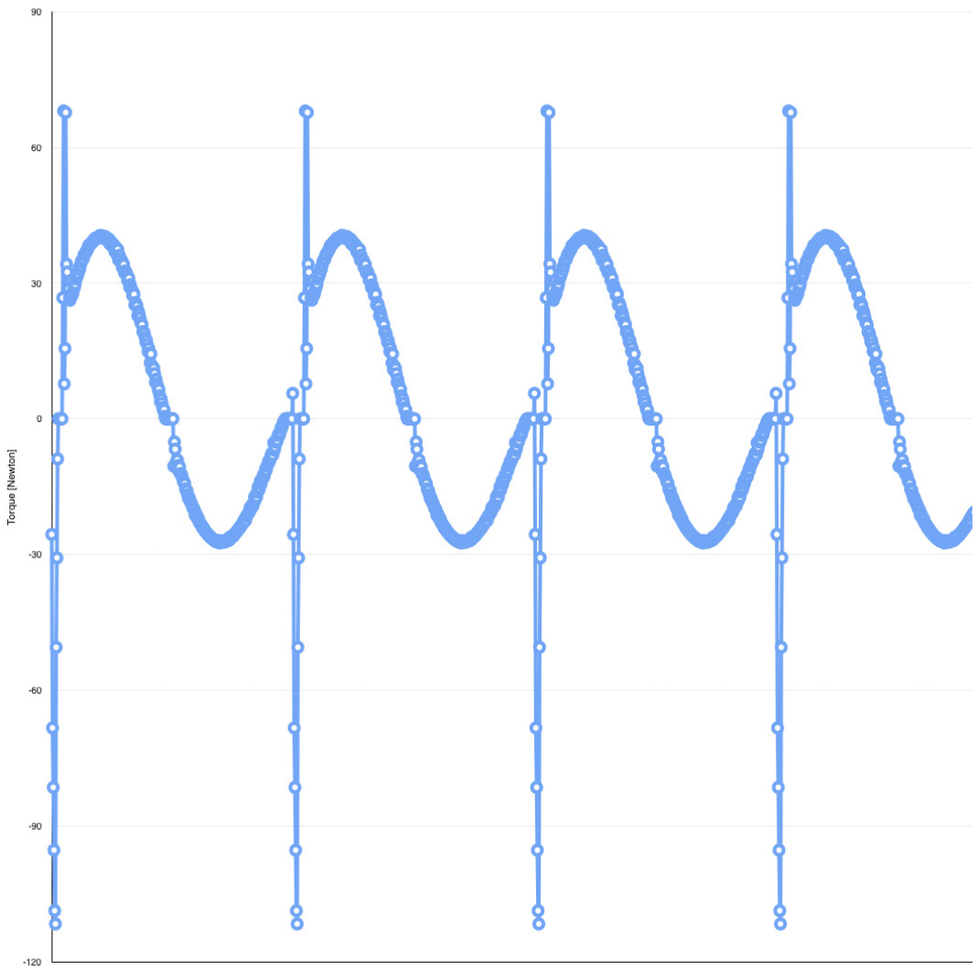


Figure 7.78: Fixed Base Torque, Kd = 100

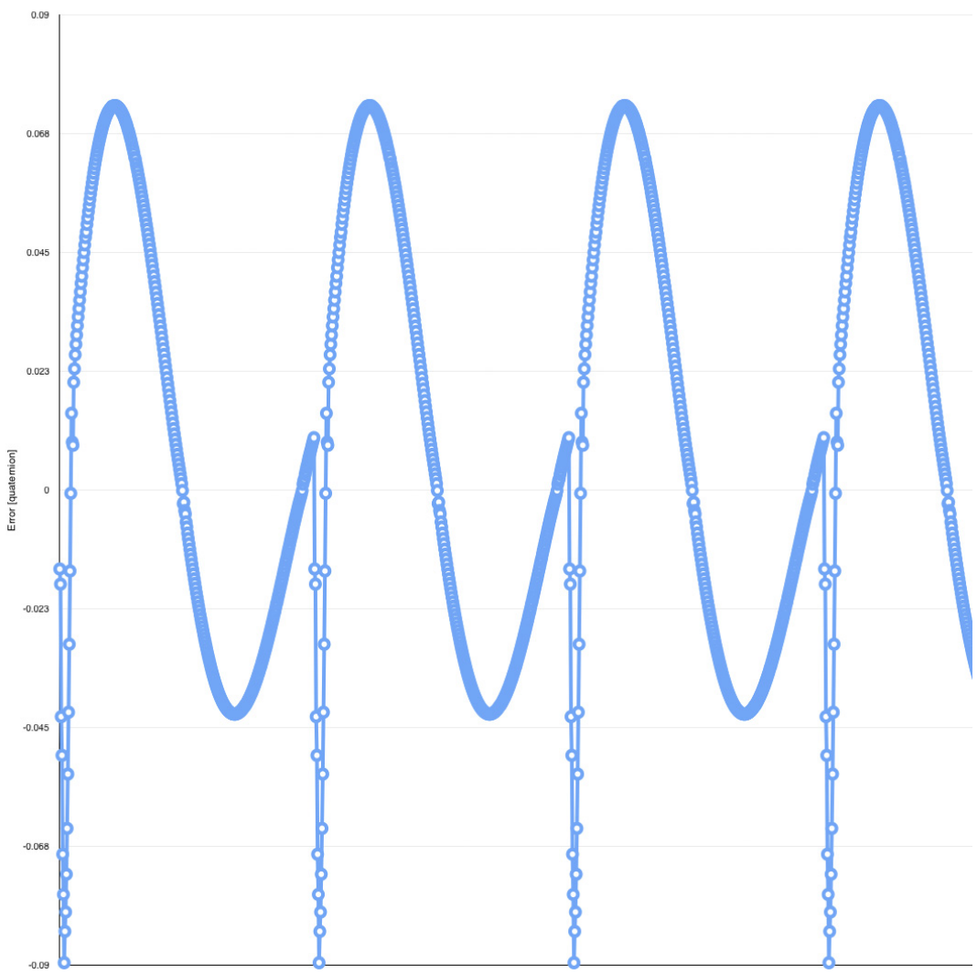


Figure 7.79: Fixed Base Position error, Kd = 150

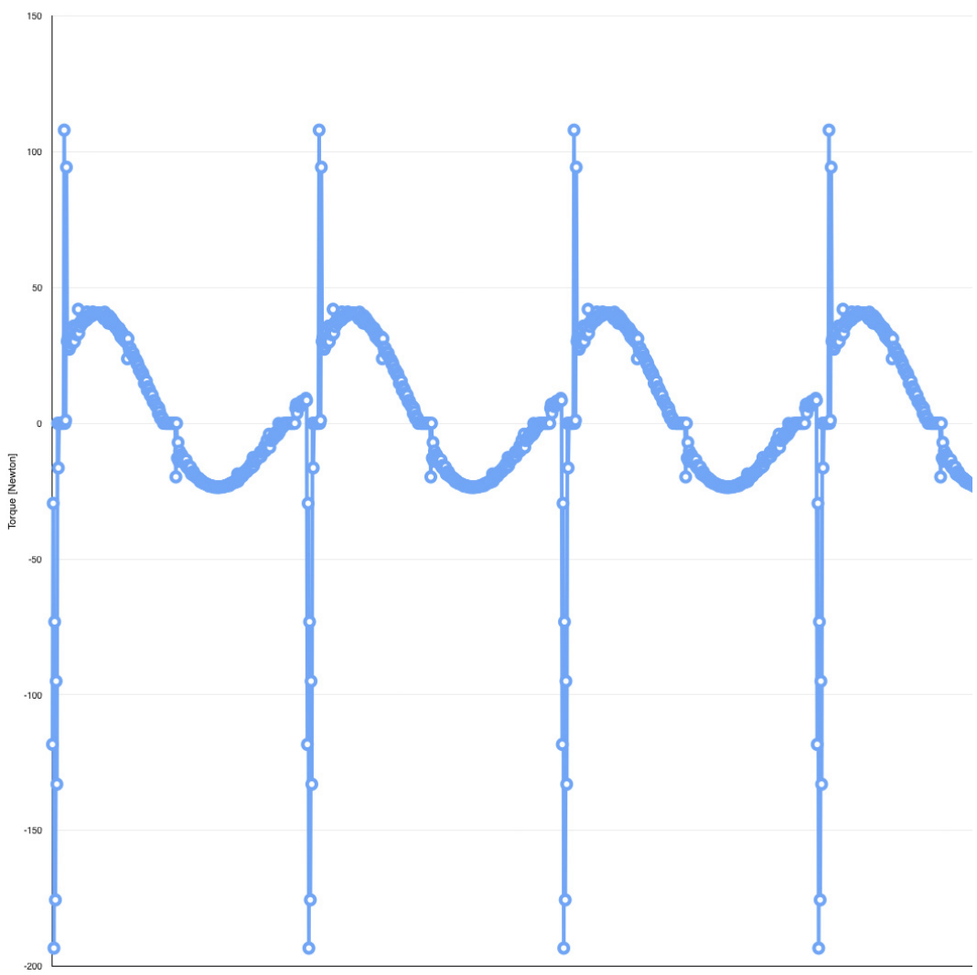


Figure 7.80: Fixed Base Torque, Kd = 150

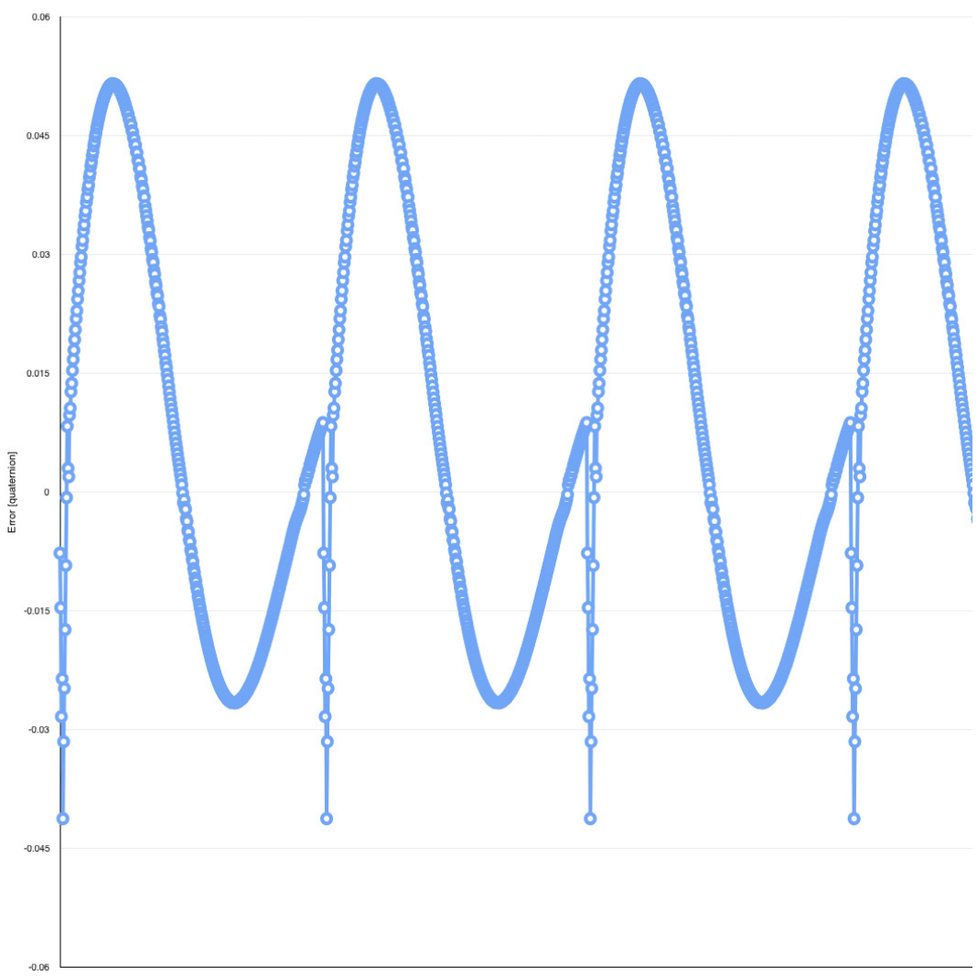


Figure 7.81: Fixed Base Position error, Kd = 200

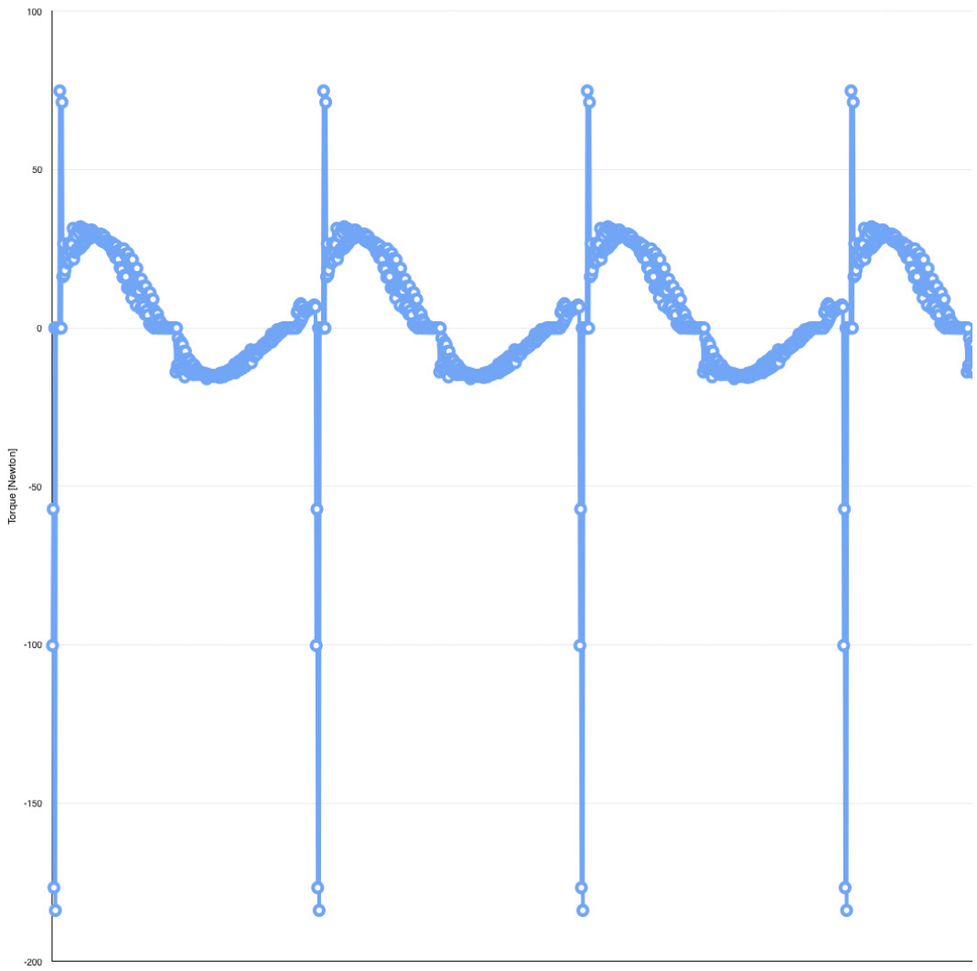


Figure 7.82: Fixed Base Torque, Kd = 200

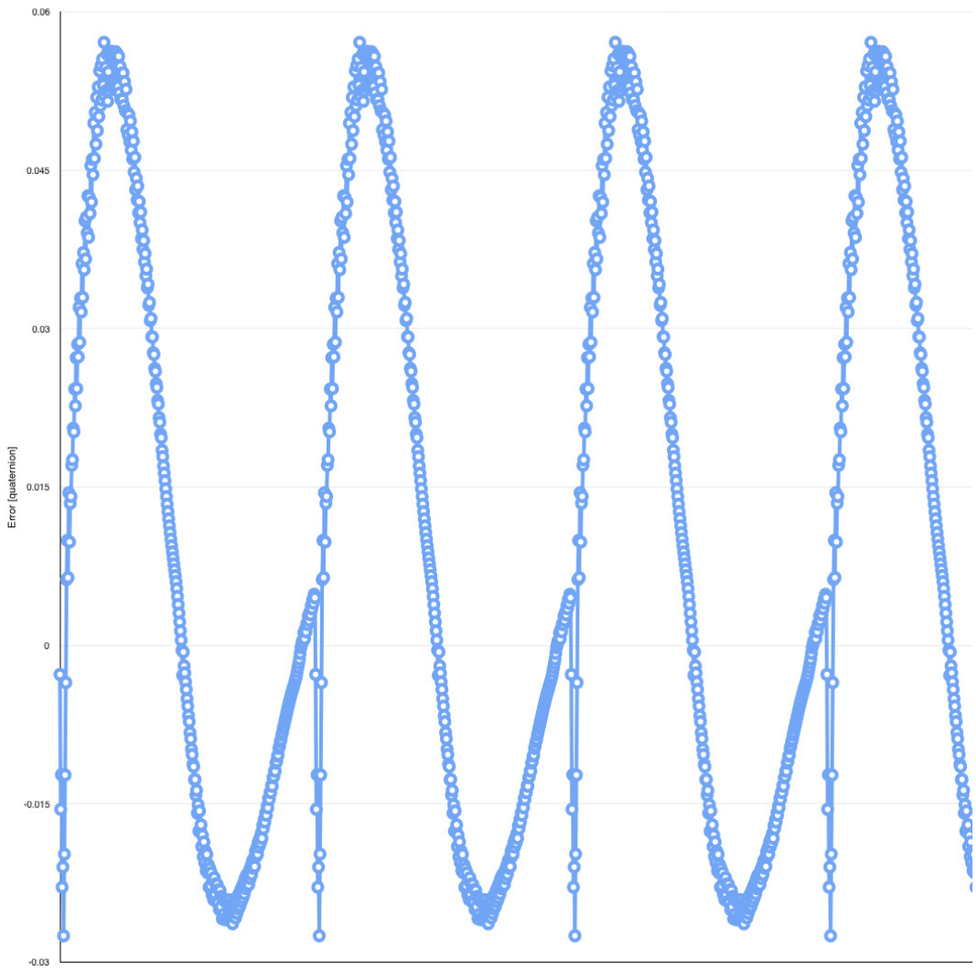


Figure 7.83: Fixed Base Position error, Kd = 250

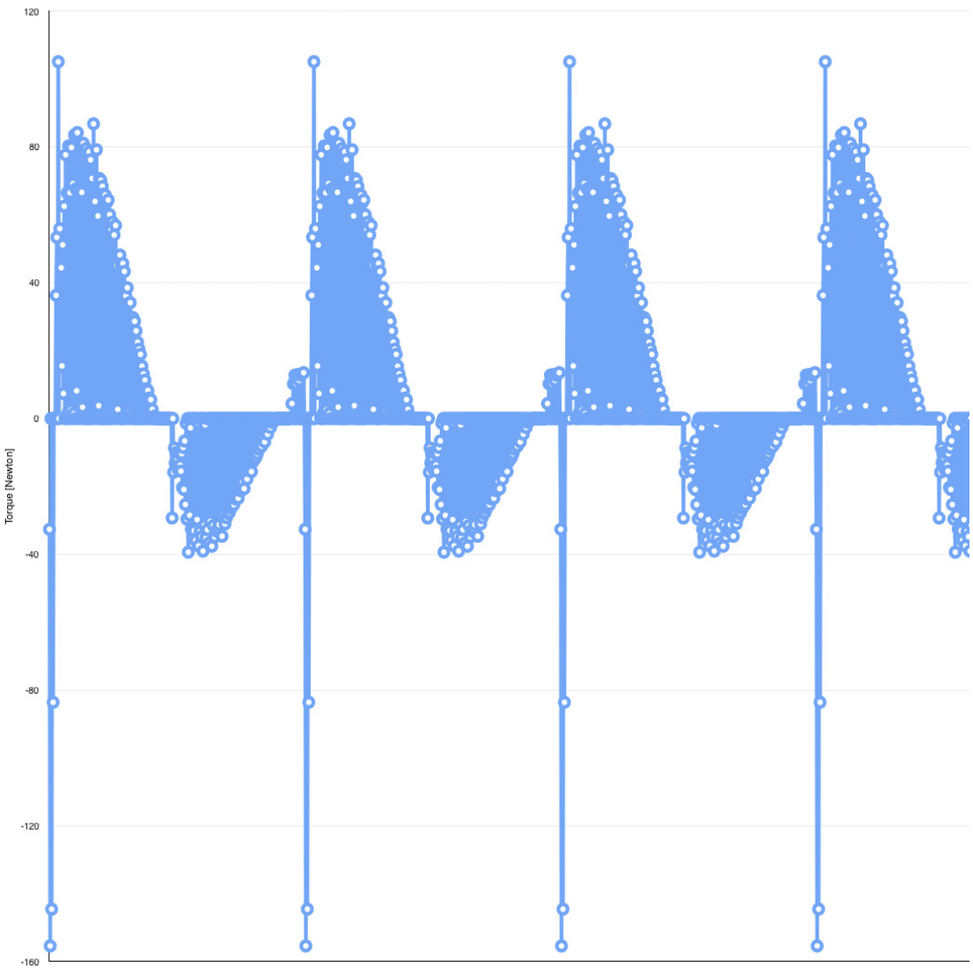


Figure 7.84: Fixed Base Torque, Kd = 250

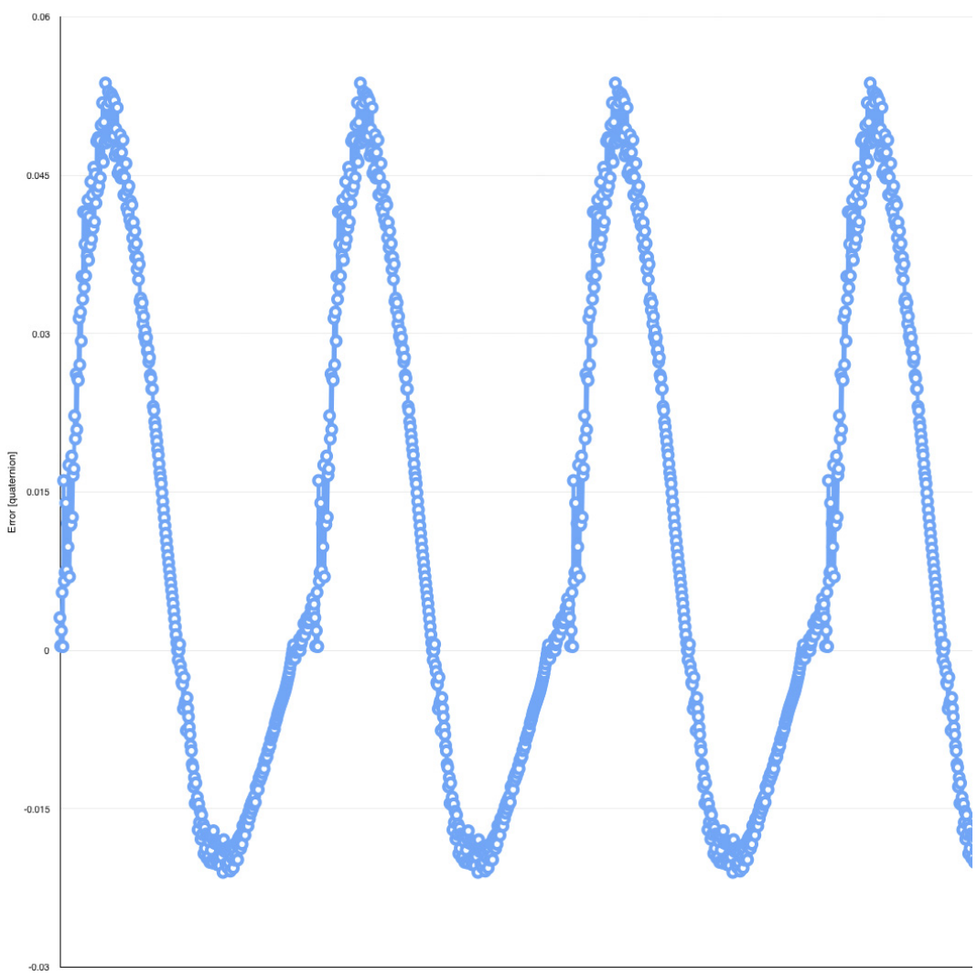


Figure 7.85: Fixed Base Position error, Kd = 300

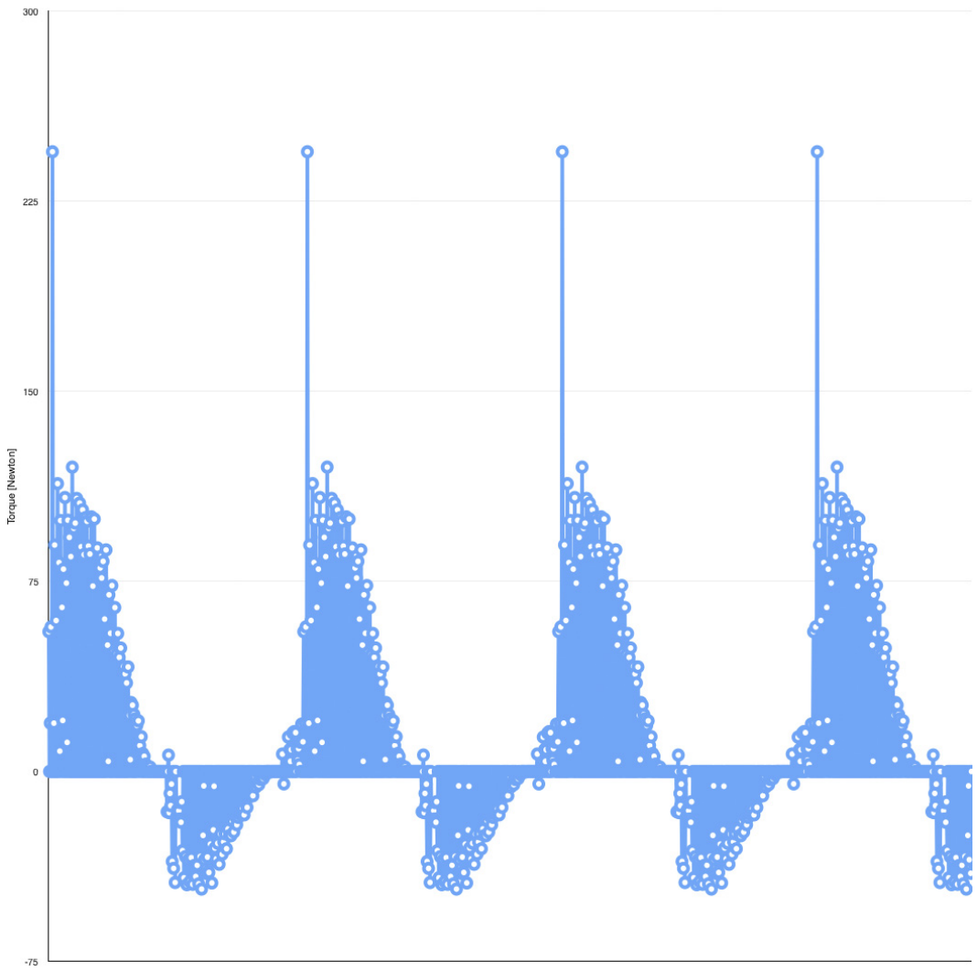


Figure 7.86: Fixed Base Torque, Kd = 300

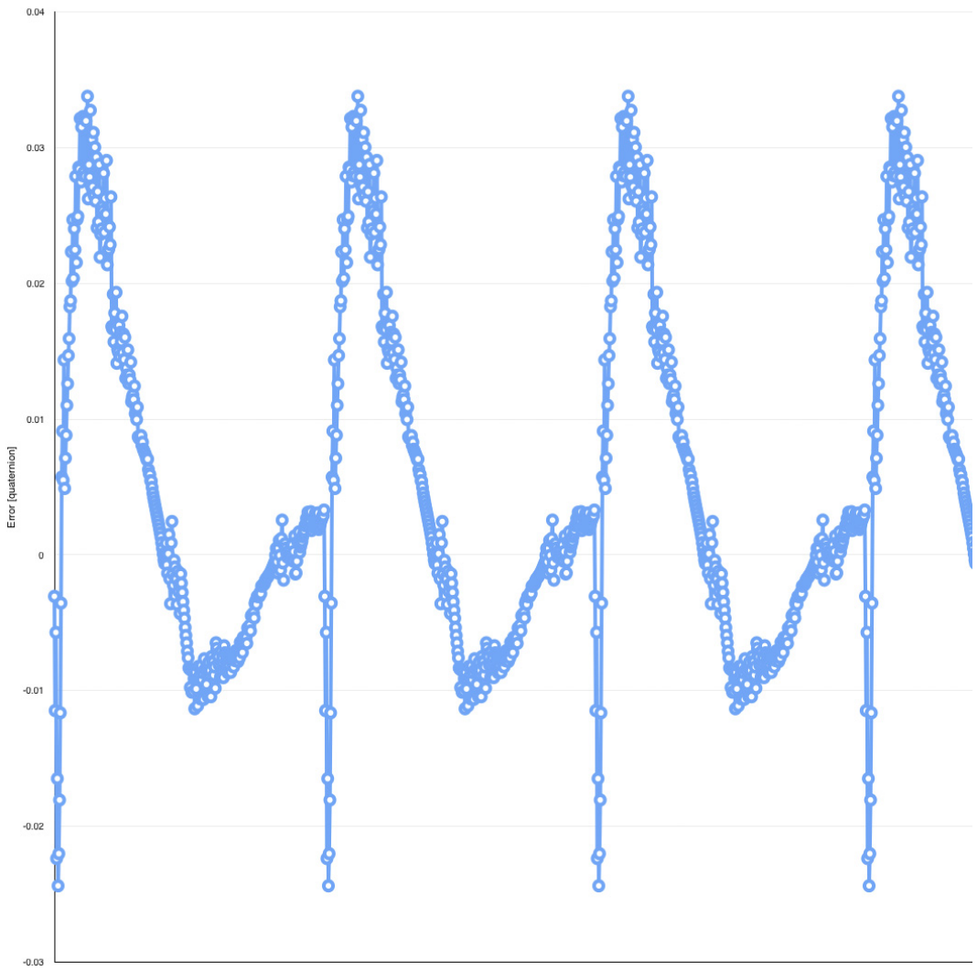


Figure 7.87: Fixed Base Position error, Kd = 350

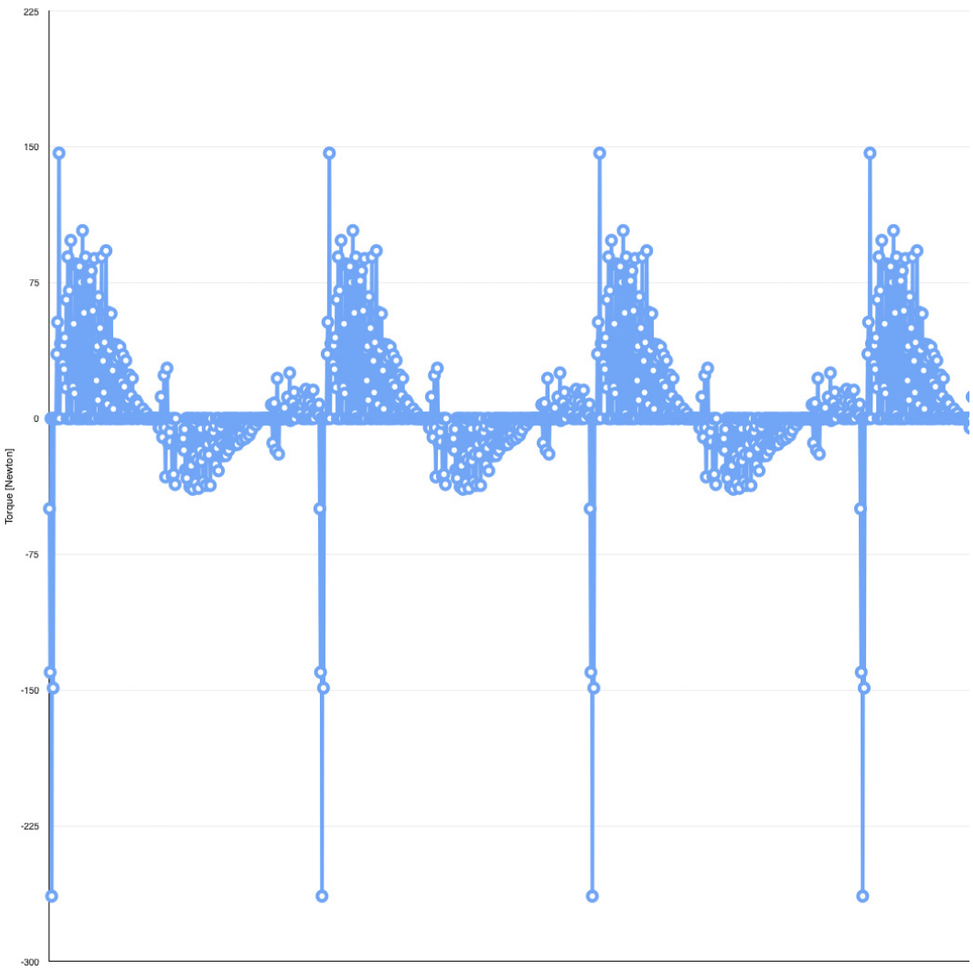


Figure 7.88: Fixed Base Torque, Kd = 350

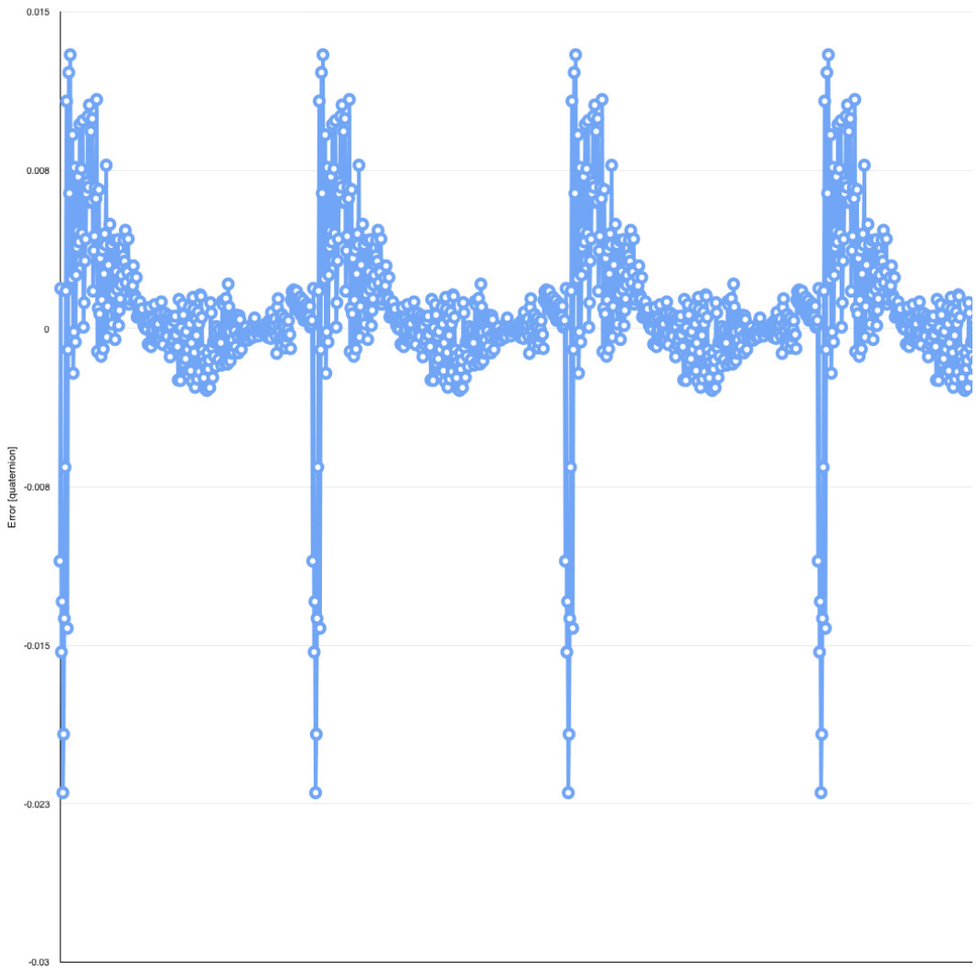


Figure 7.89: Fixed Base Position error, Kd = 400

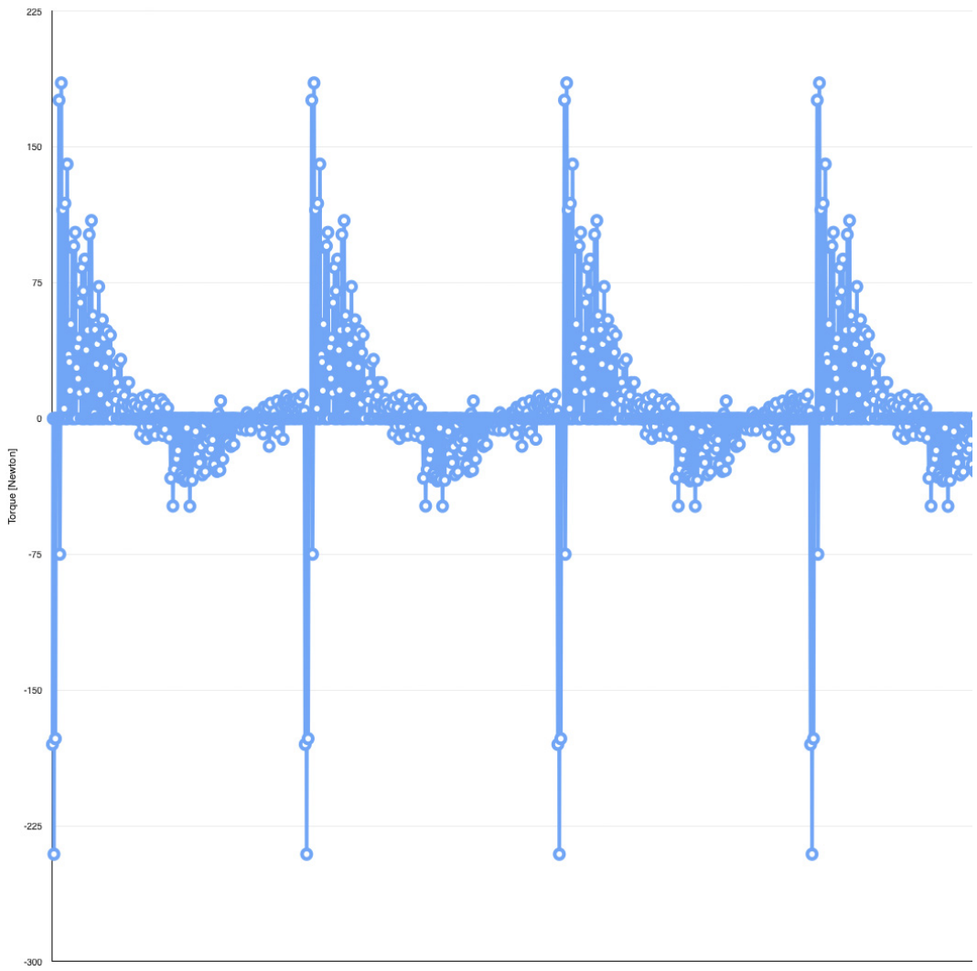


Figure 7.90: Fixed Base Torque, Kd = 400

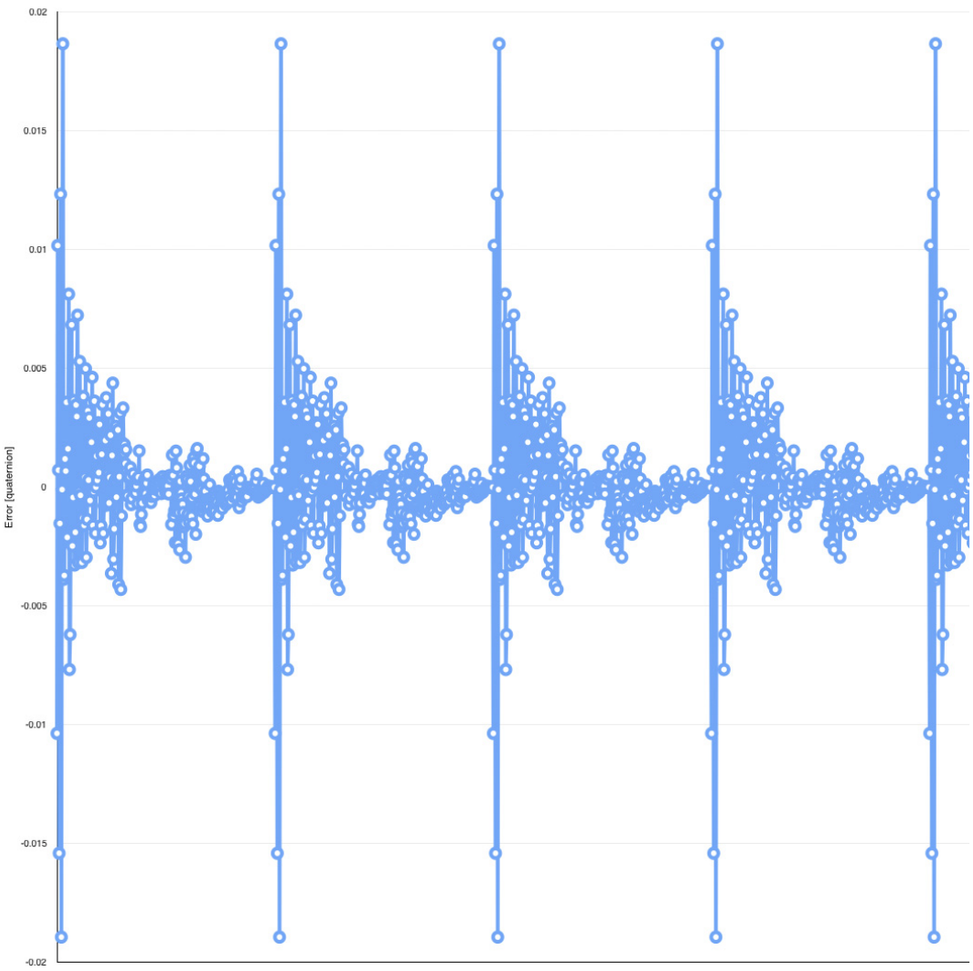


Figure 7.91: Fixed Base Position error, Kd = 450

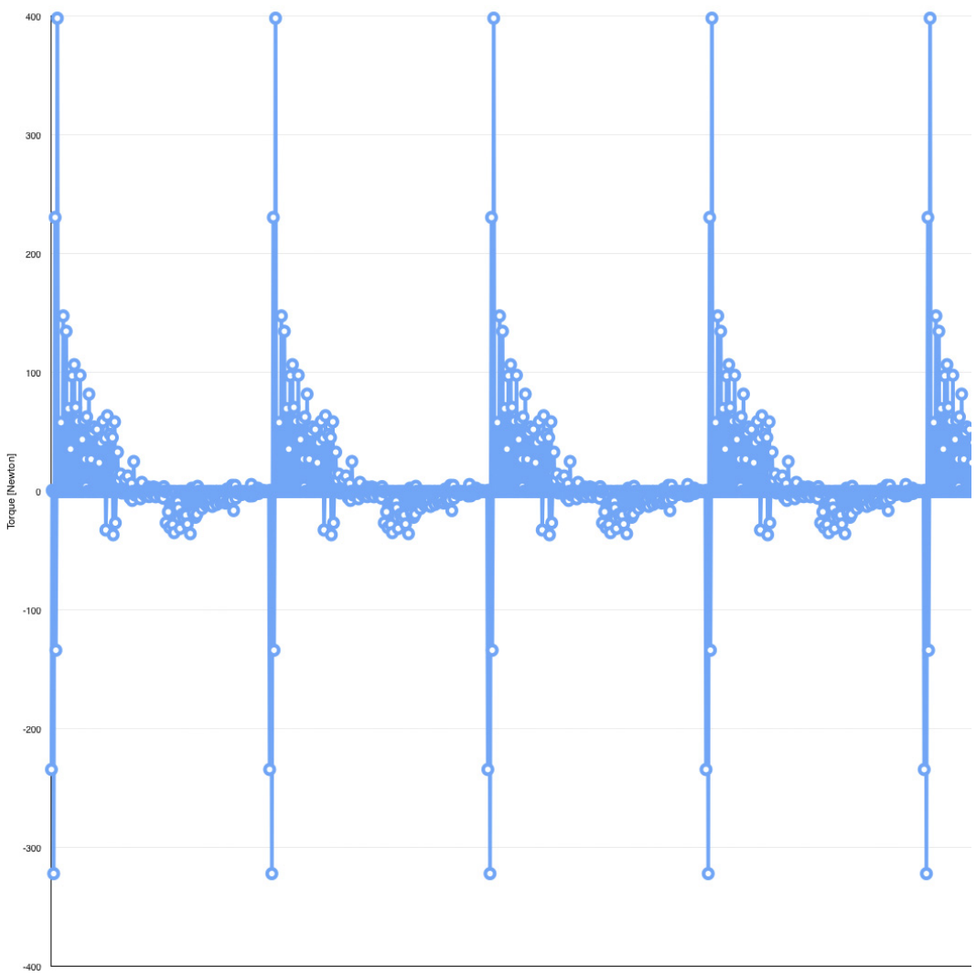


Figure 7.92: Fixed Base Torque, Kd = 450

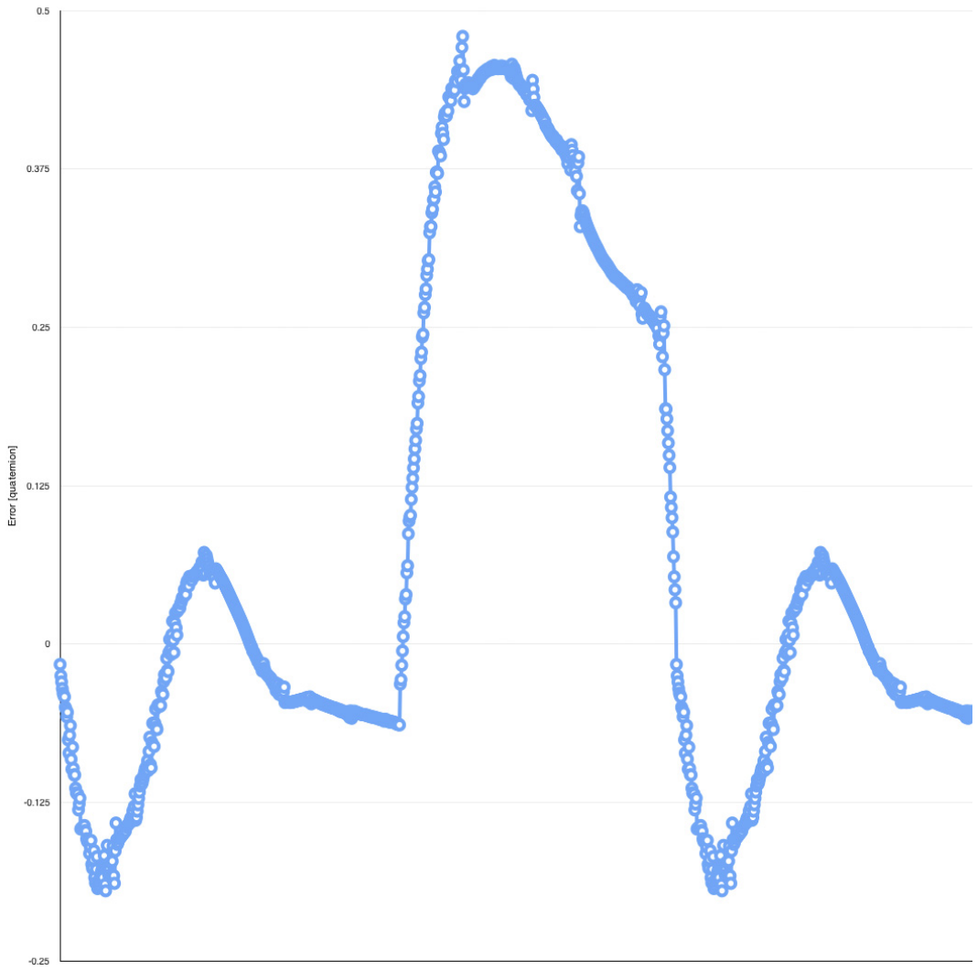


Figure 7.93: Uniform Movement Base Position error, kp = 100

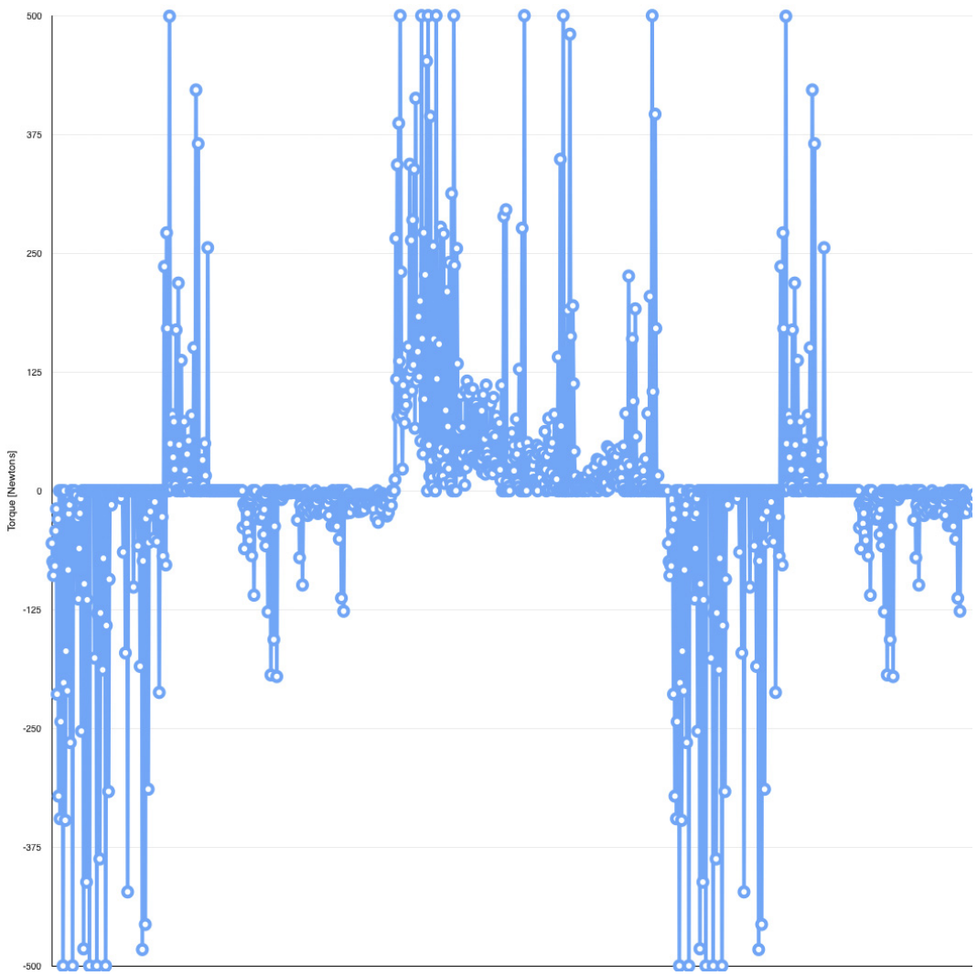


Figure 7.94: Uniform Movement Base Torque, kp = 100

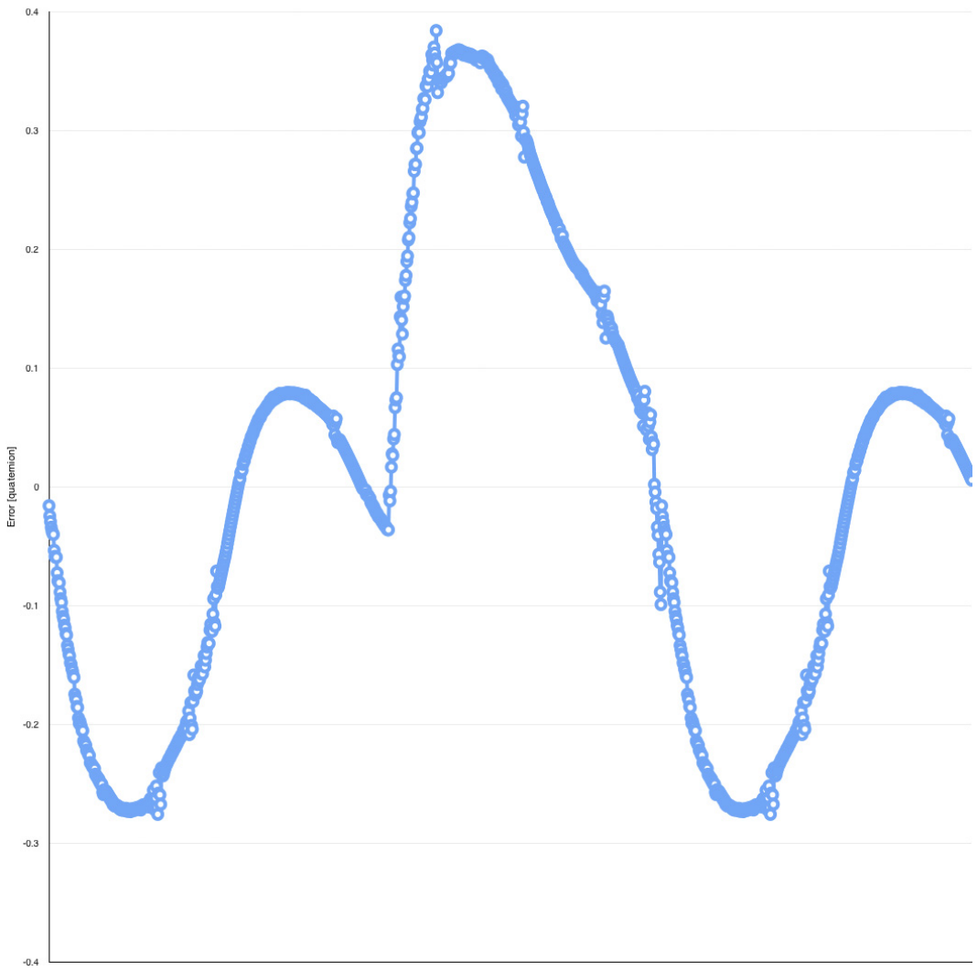


Figure 7.95: Uniform Movement Base Position error, kp = 150

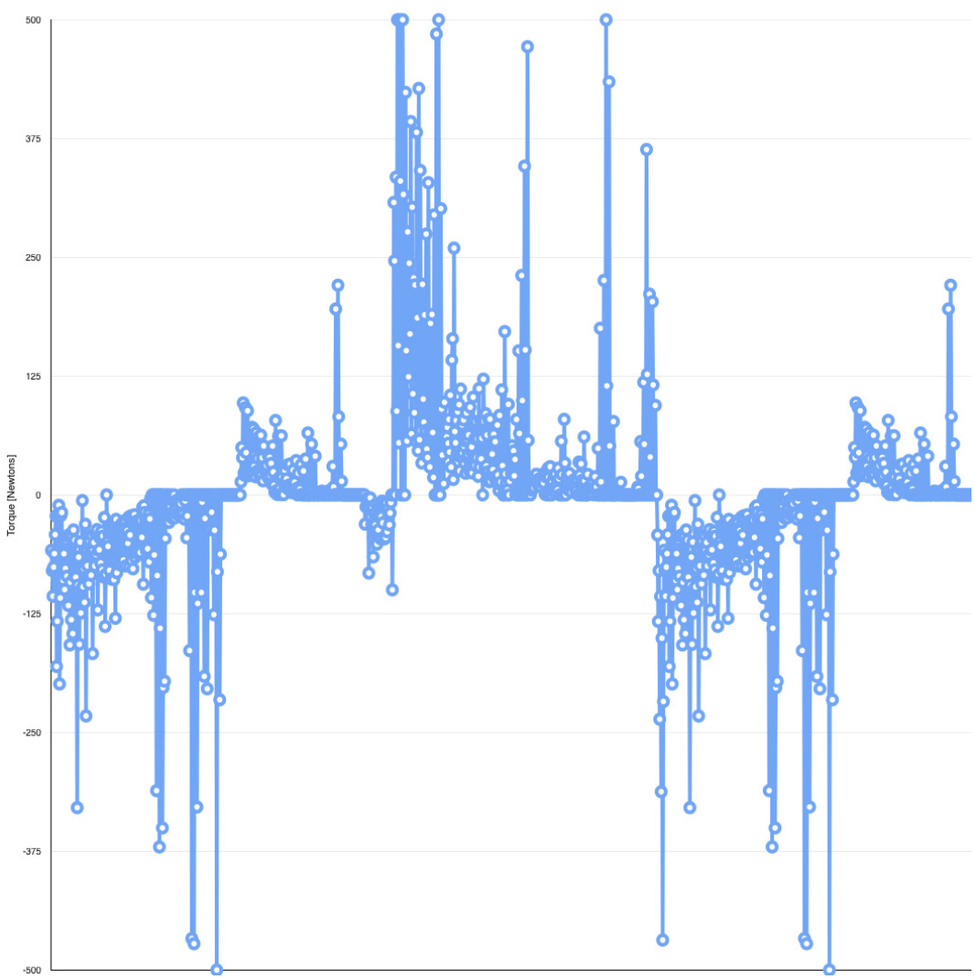


Figure 7.96: Uniform Movement Base Torque, kp = 150

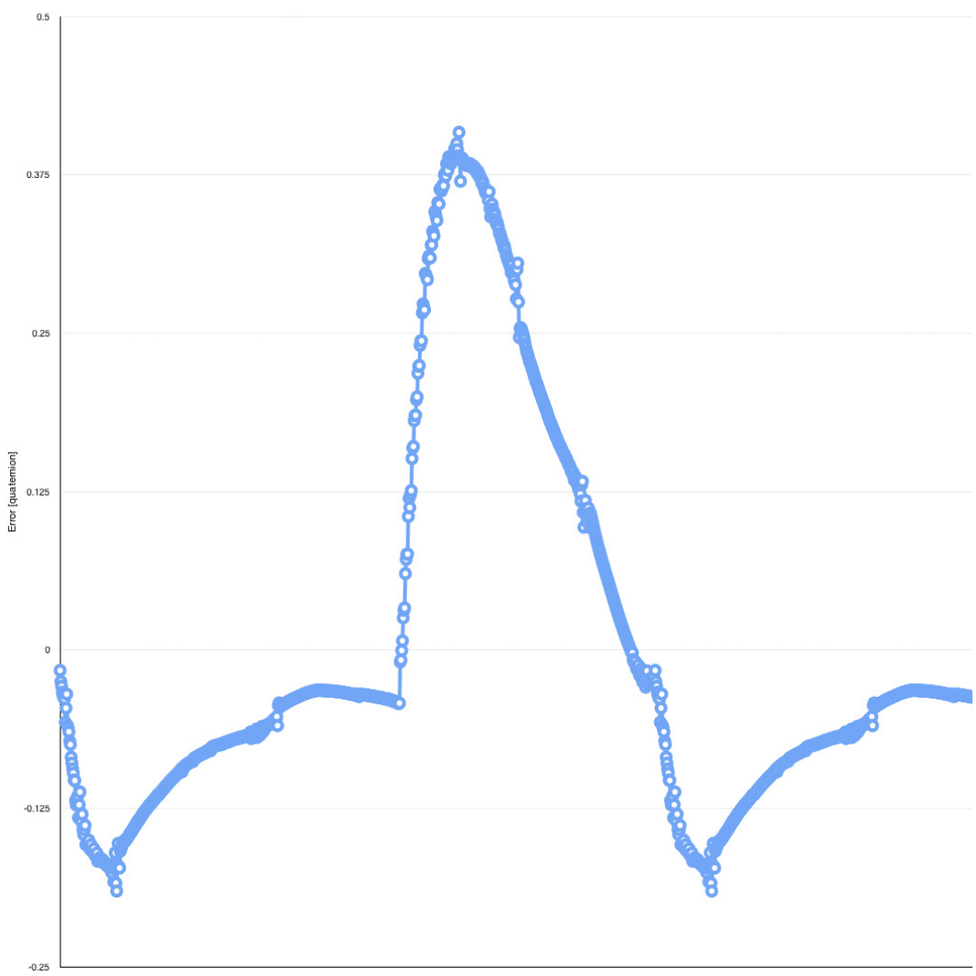


Figure 7.97: Uniform Movement Base Position error, kp = 200

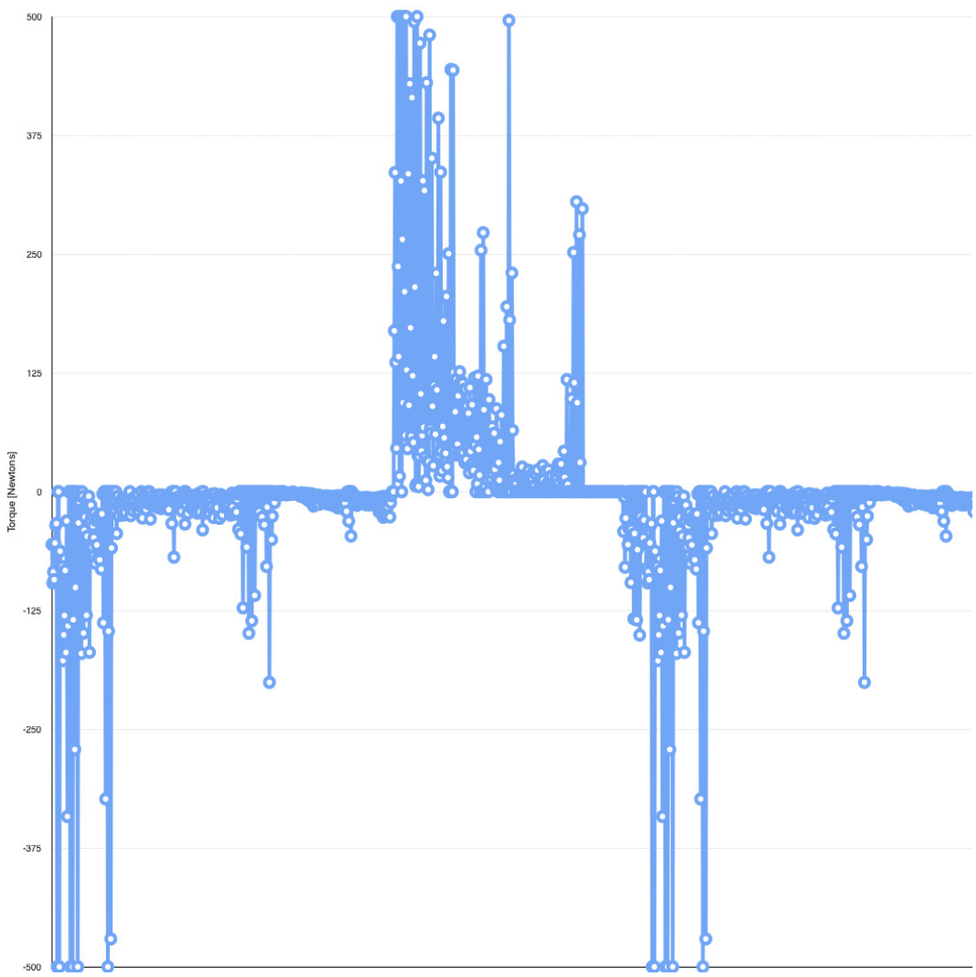


Figure 7.98: Uniform Movement Base Torque, kp = 200

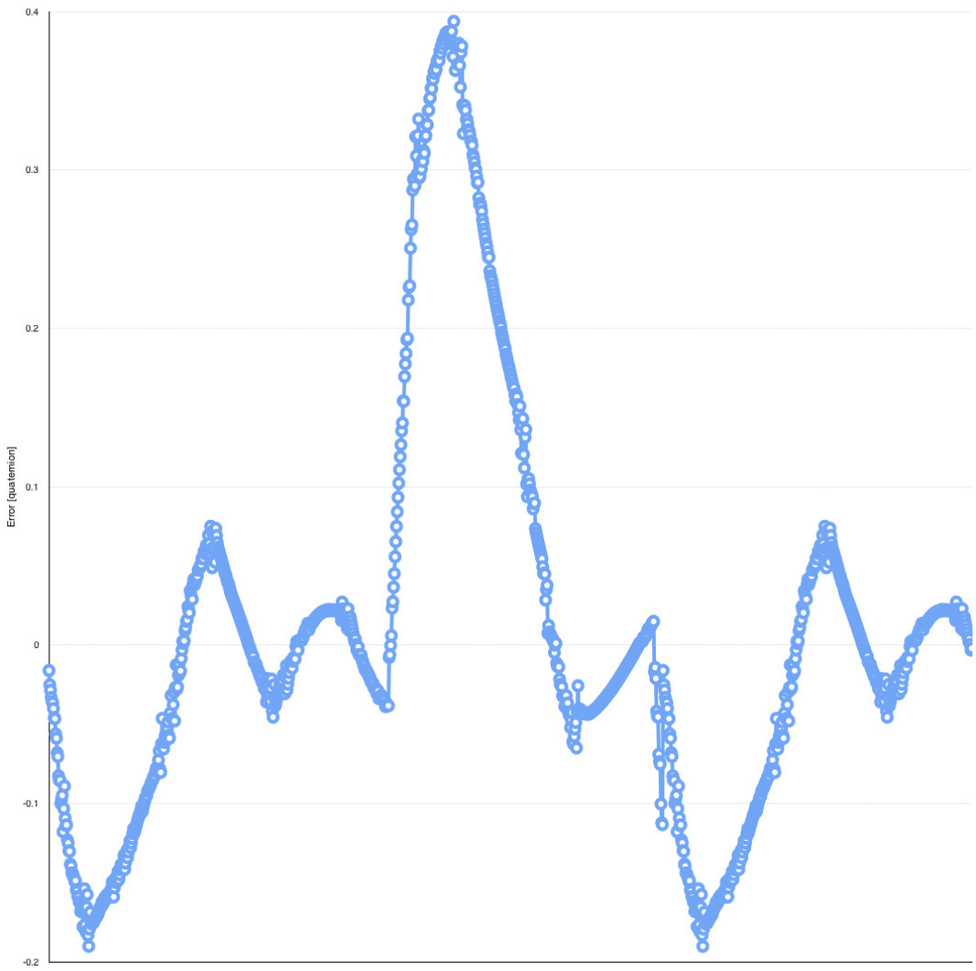


Figure 7.99: Uniform Movement Base Position error, kp = 250

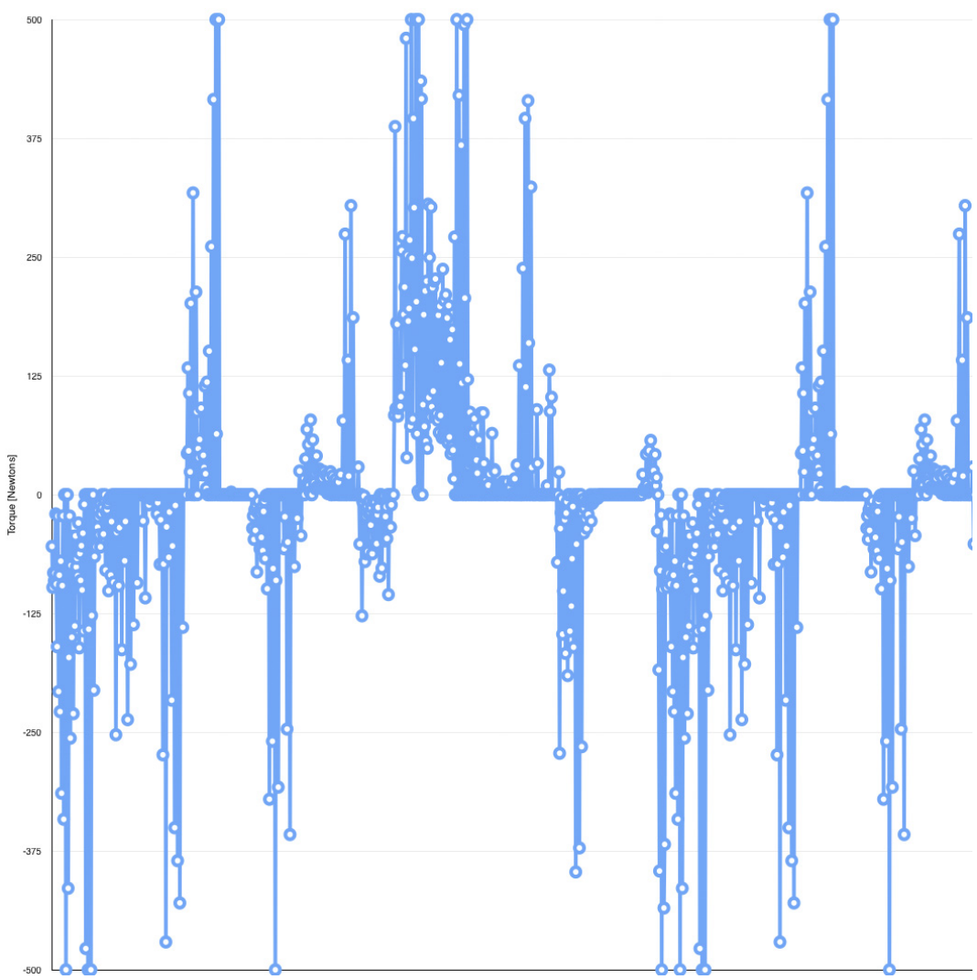


Figure 7.100: Uniform Movement Base Torque, kp = 250

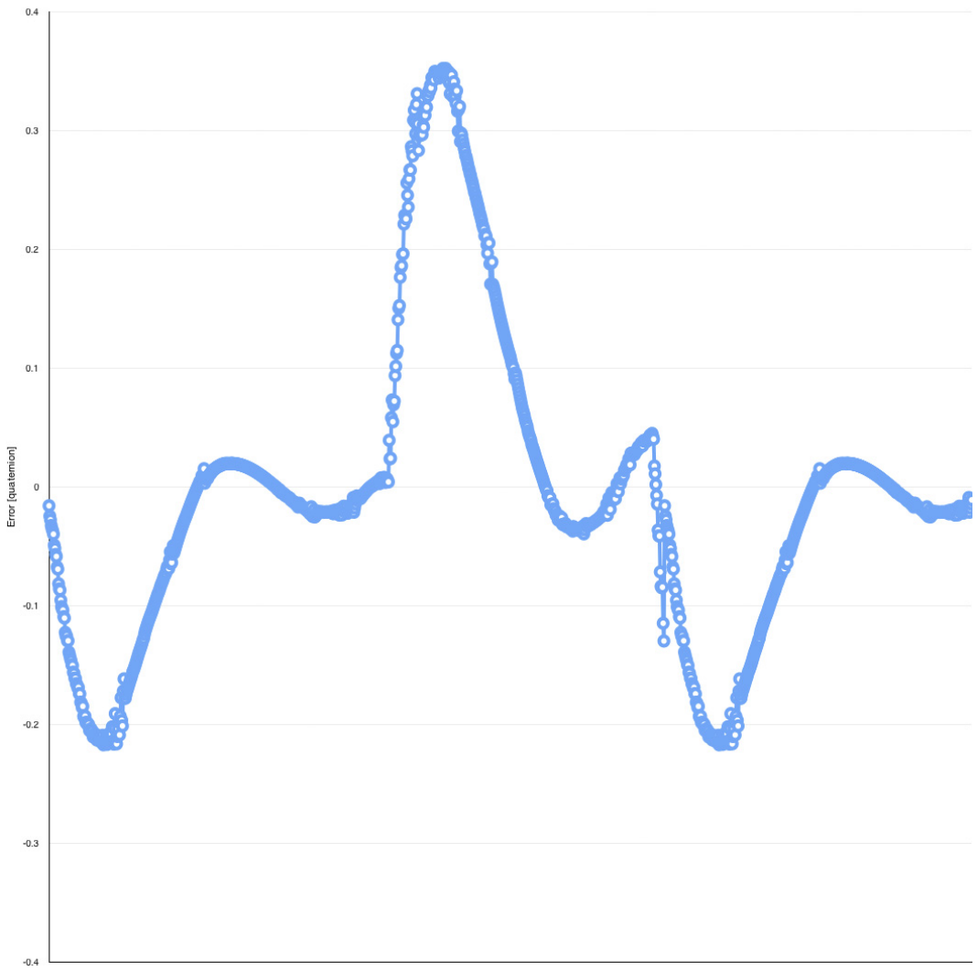


Figure 7.101: Uniform Movement Base Position error, kp = 300

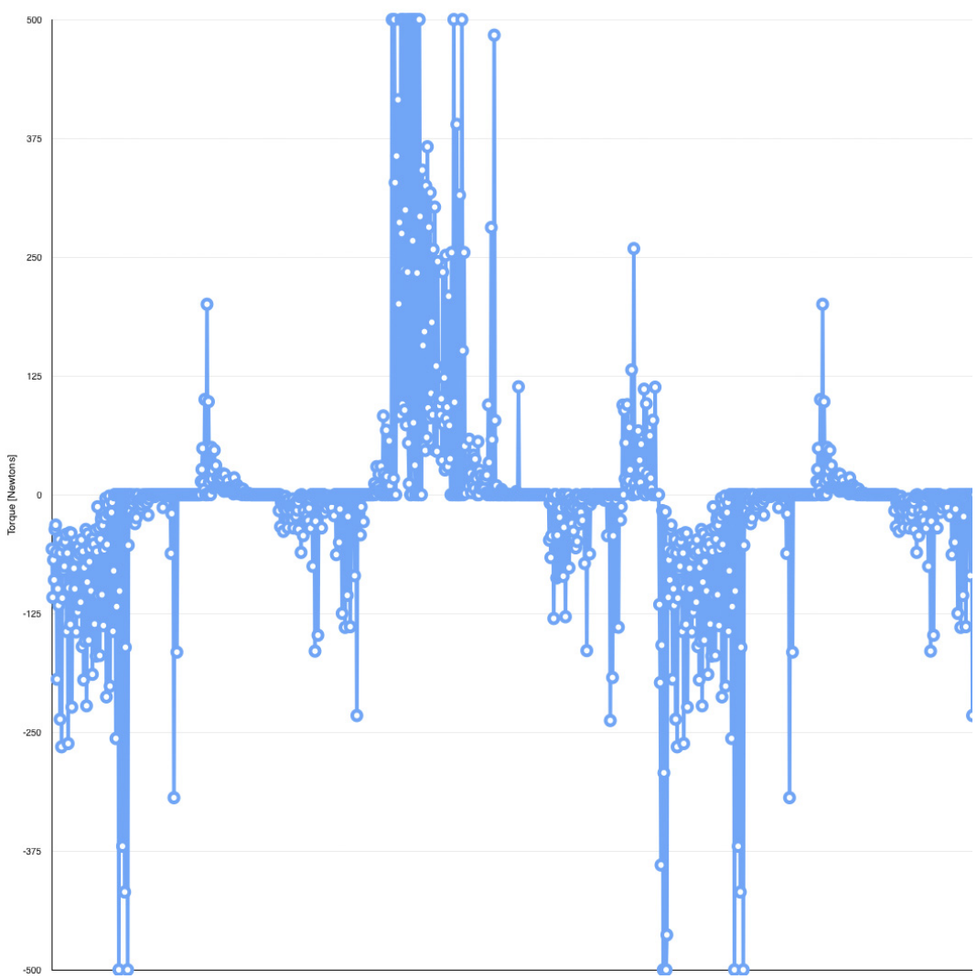


Figure 7.102: Uniform Movement Base Torque, kp = 300

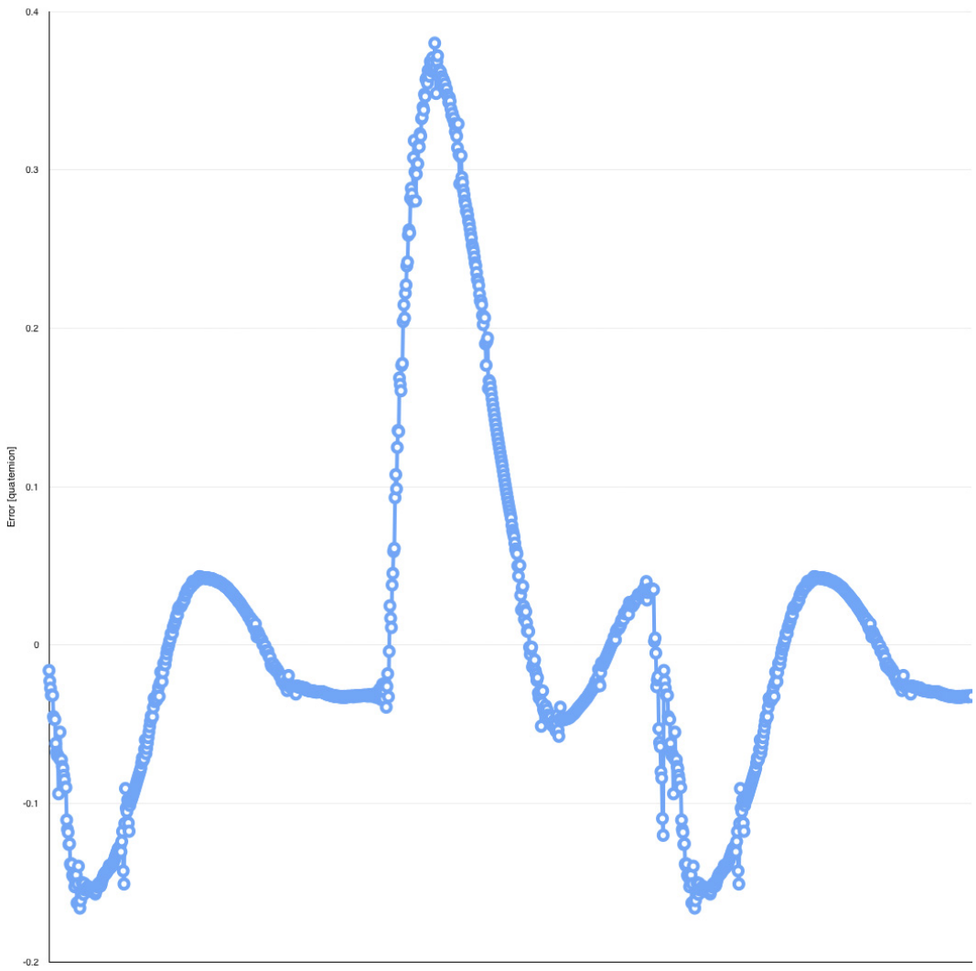


Figure 7.103: Uniform Movement Base Position error, kp = 350

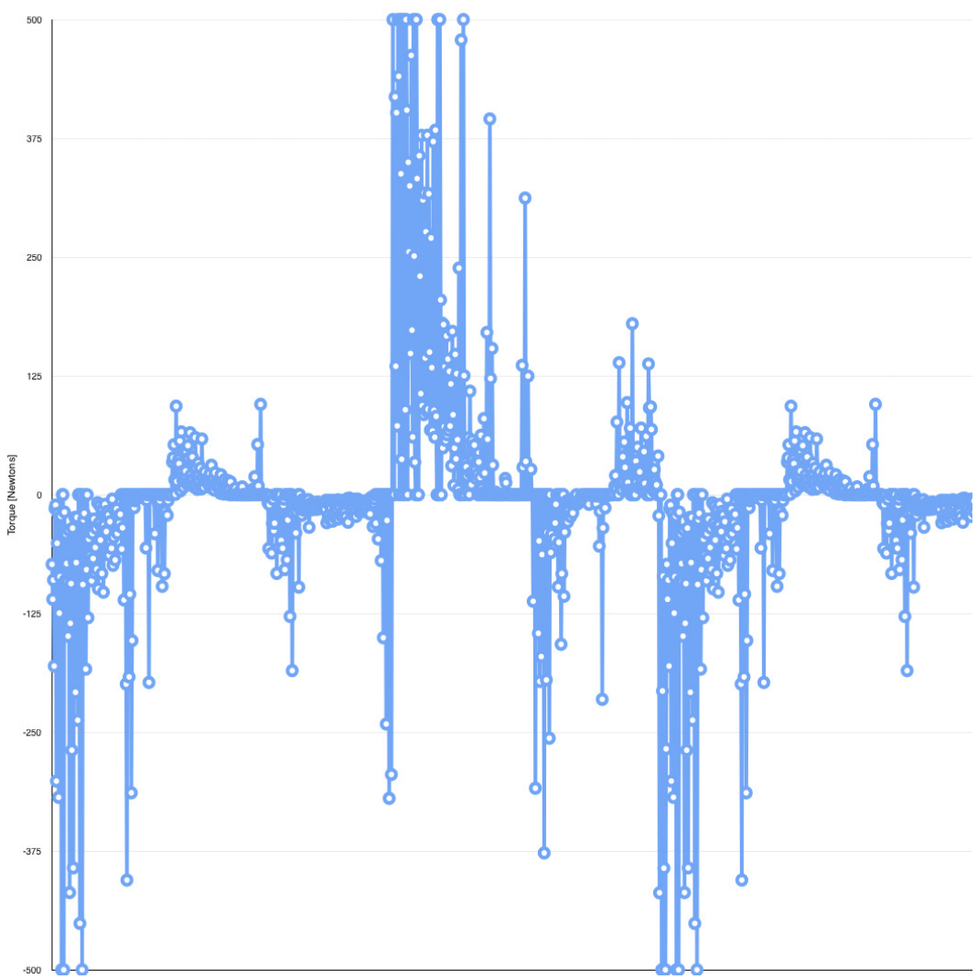


Figure 7.104: Uniform Movement Base Torque, kp = 350

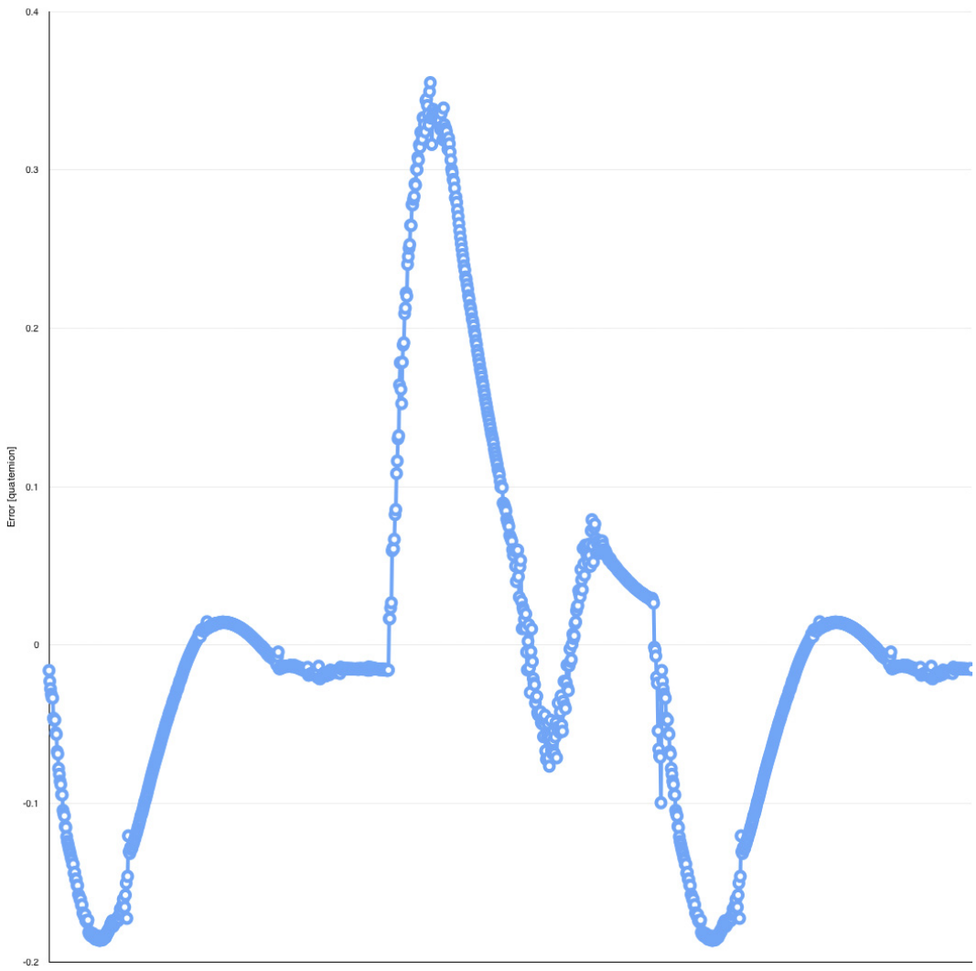


Figure 7.105: Uniform Movement Base Position error, kp = 400

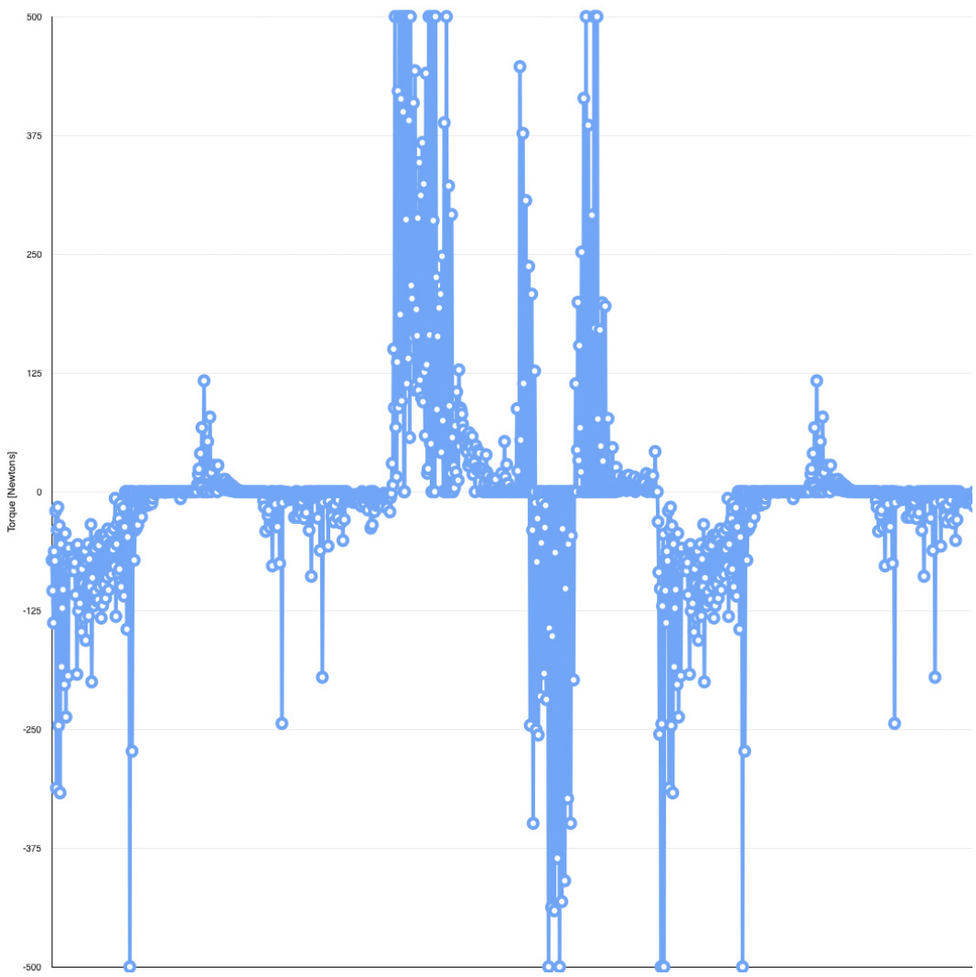


Figure 7.106: Uniform Movement Base Torque, kp = 400

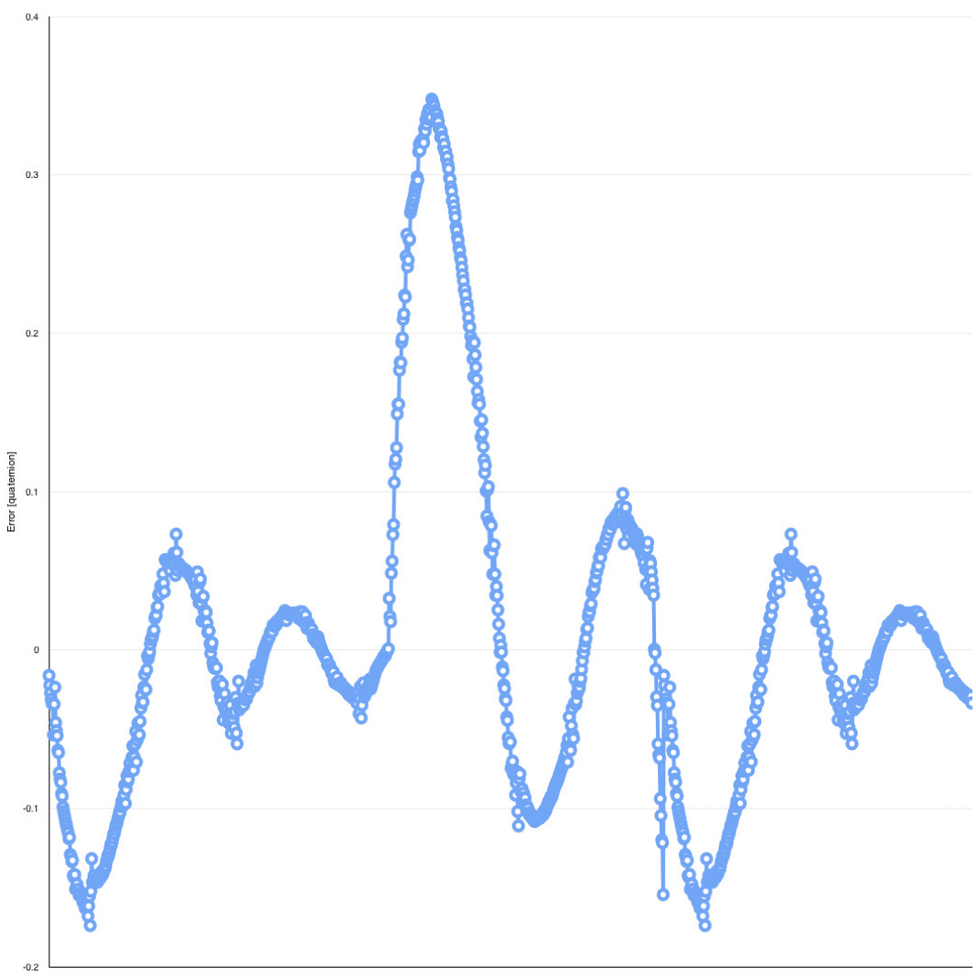


Figure 7.107: Uniform Movement Base Position error, kp = 500

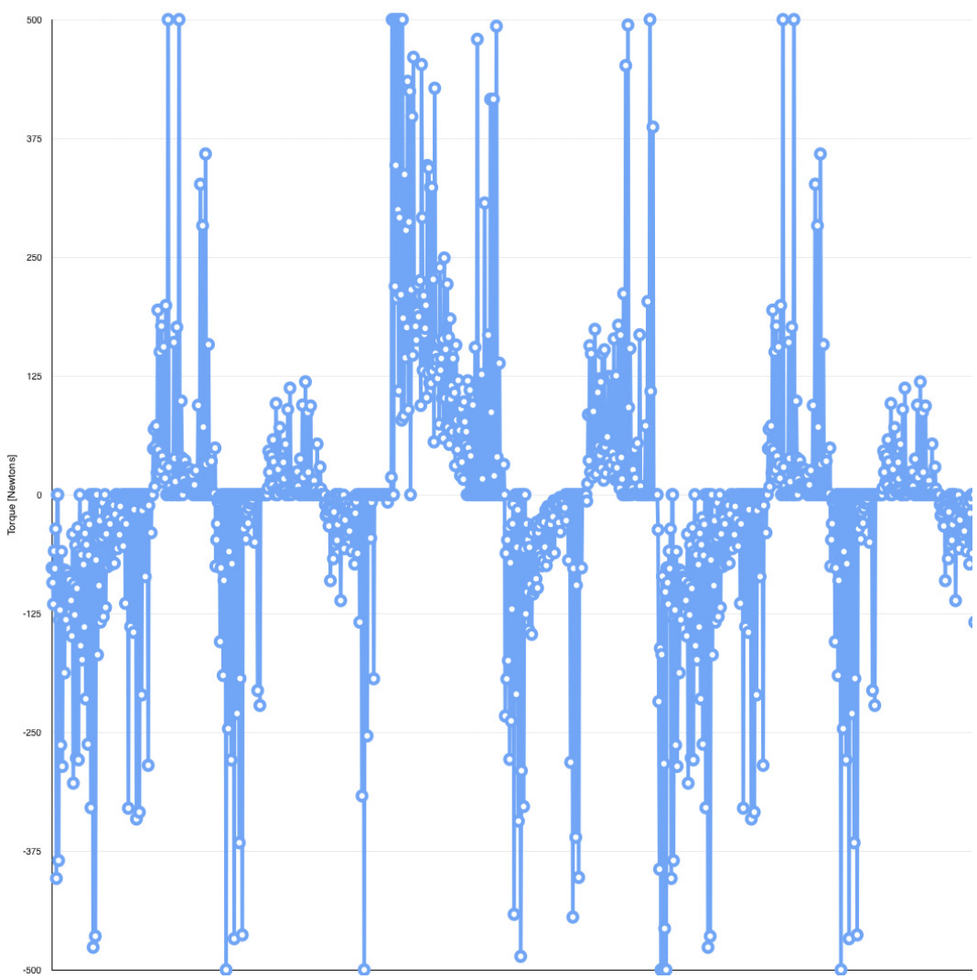


Figure 7.108: Uniform Movement Base Torque, kp = 500

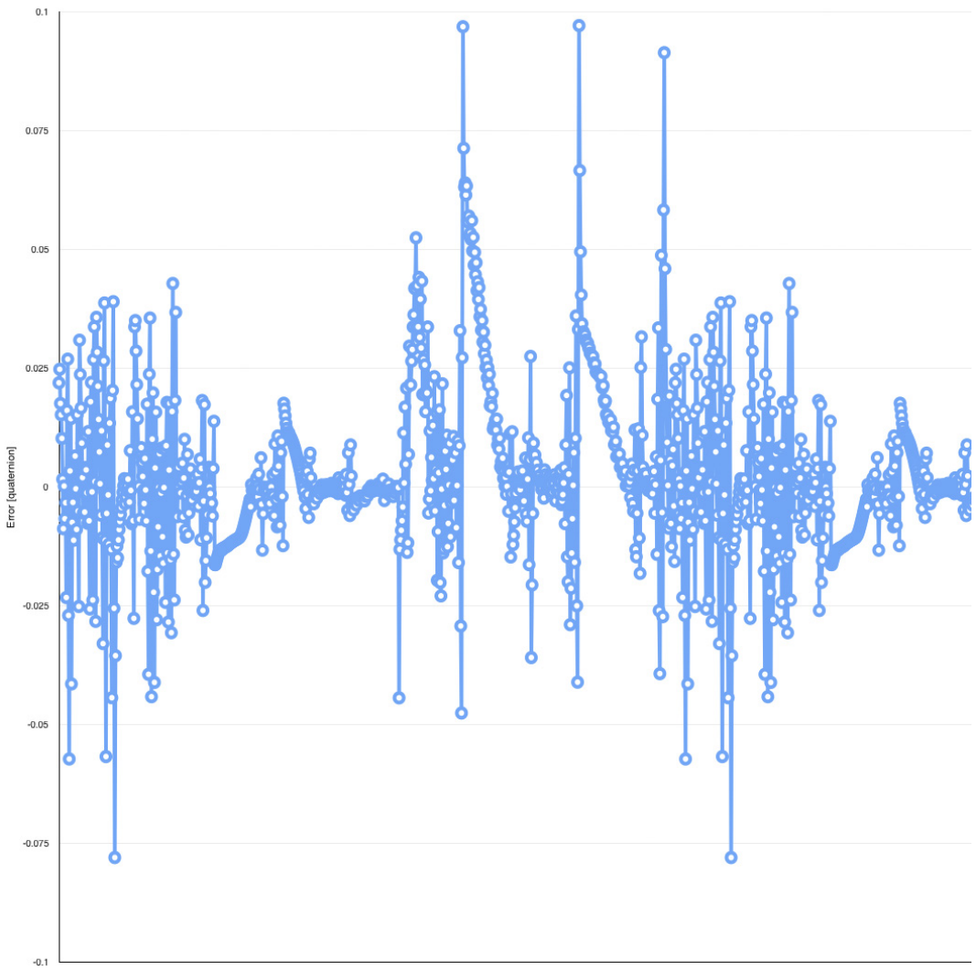


Figure 7.109: Uniform Movement Base Position error, kp = 100

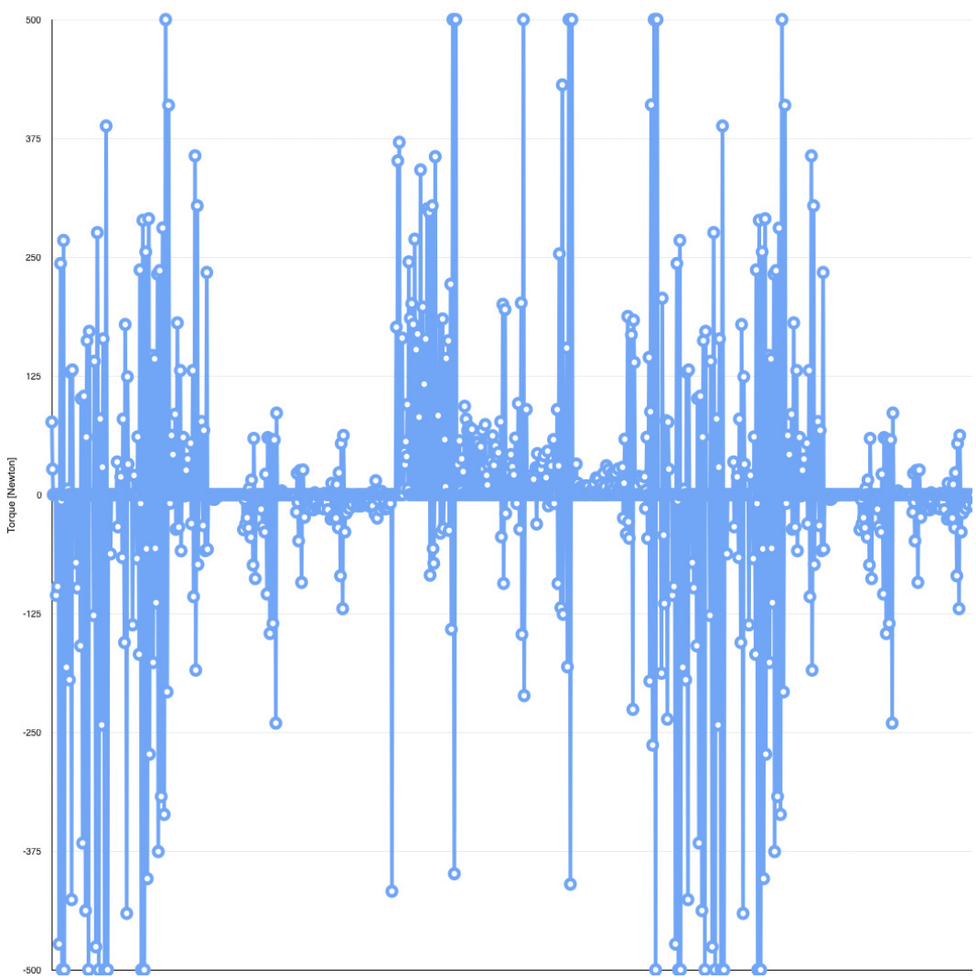


Figure 7.110: Uniform Movement Base Torque, kp = 100

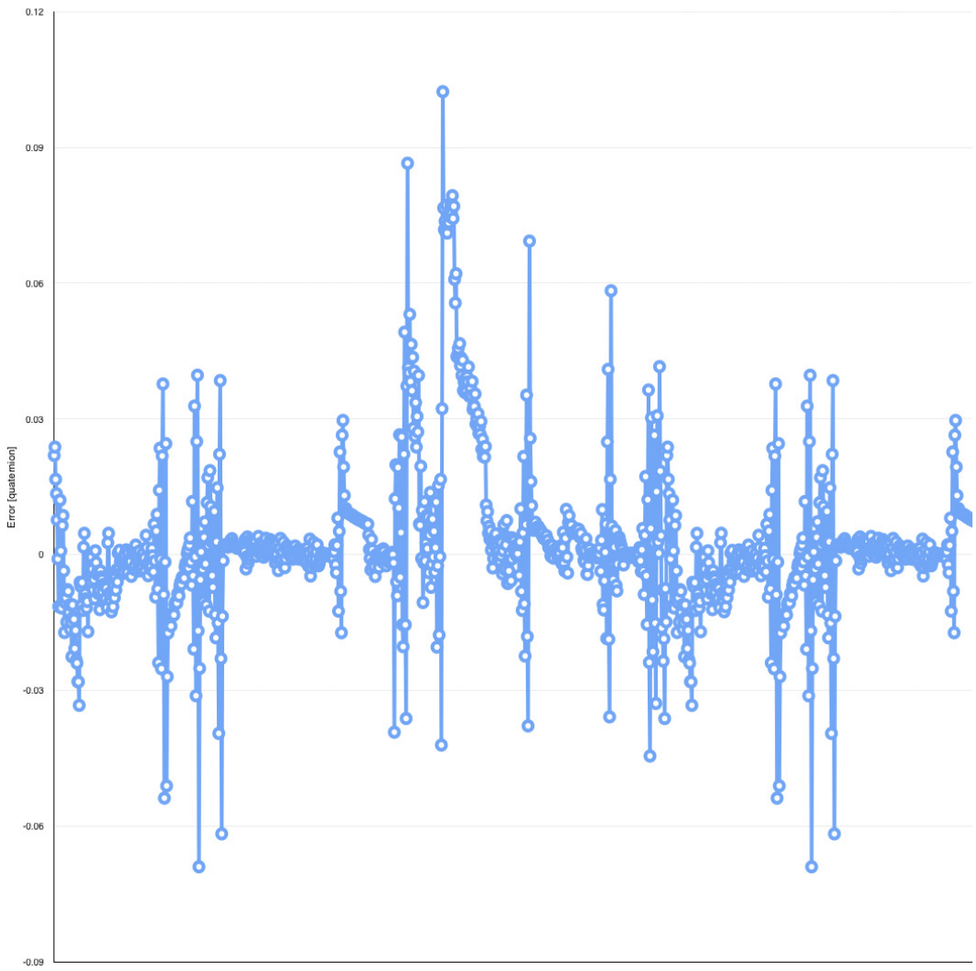


Figure 7.111: Uniform Movement Base Position error, kp = 150

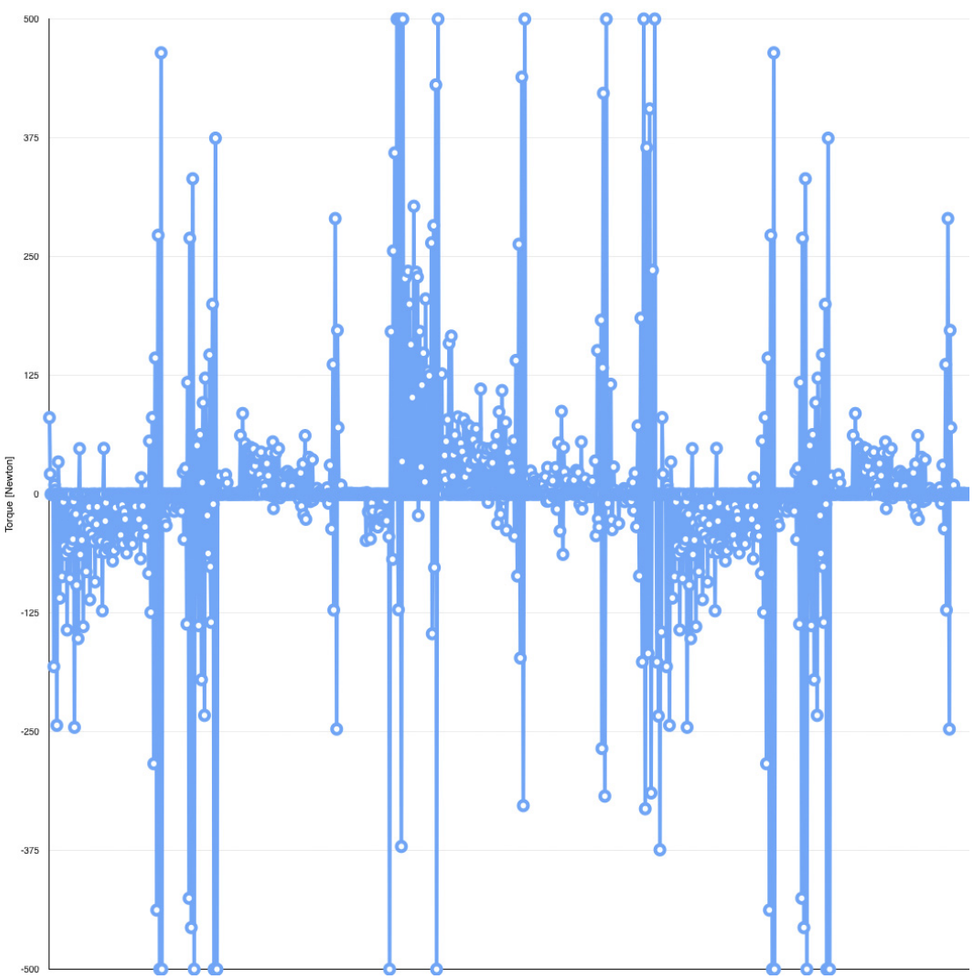


Figure 7.112: Uniform Movement Base Torque, kp = 150

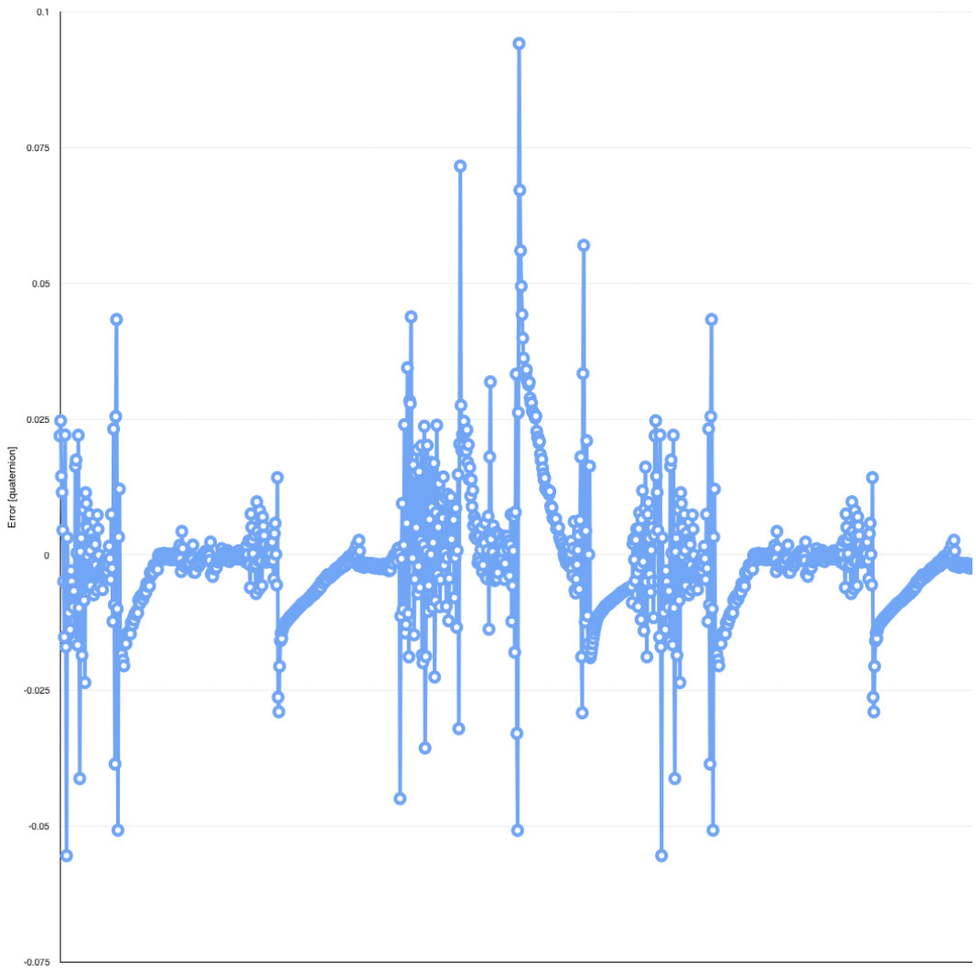


Figure 7.113: Uniform Movement Base Position error, kp = 200

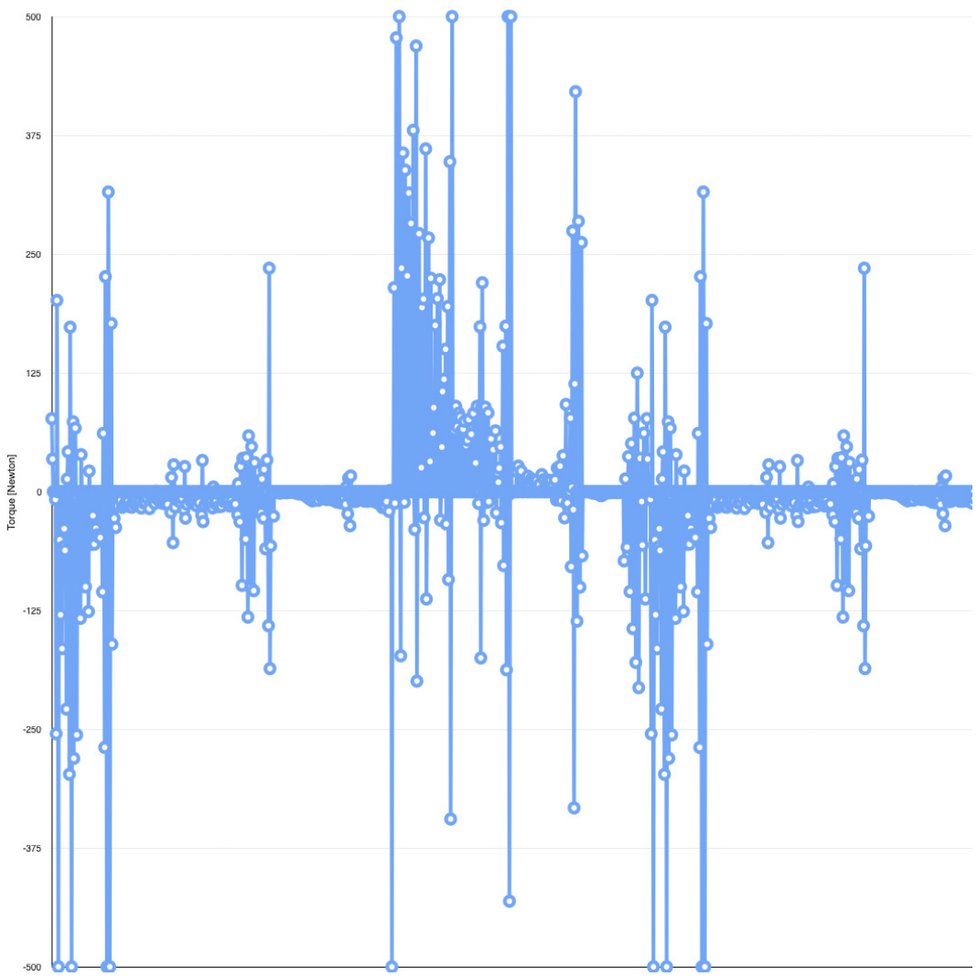


Figure 7.114: Uniform Movement Base Torque, kp = 200

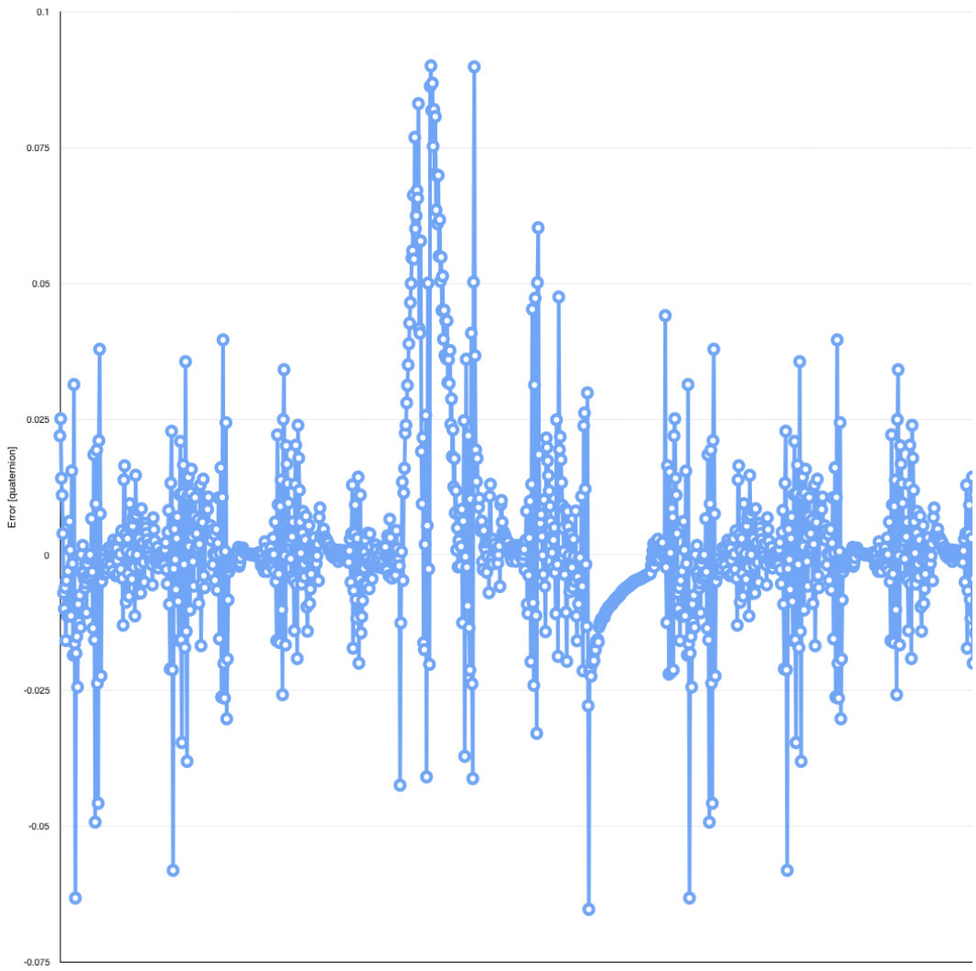


Figure 7.115: Uniform Movement Base Position error, kp = 250

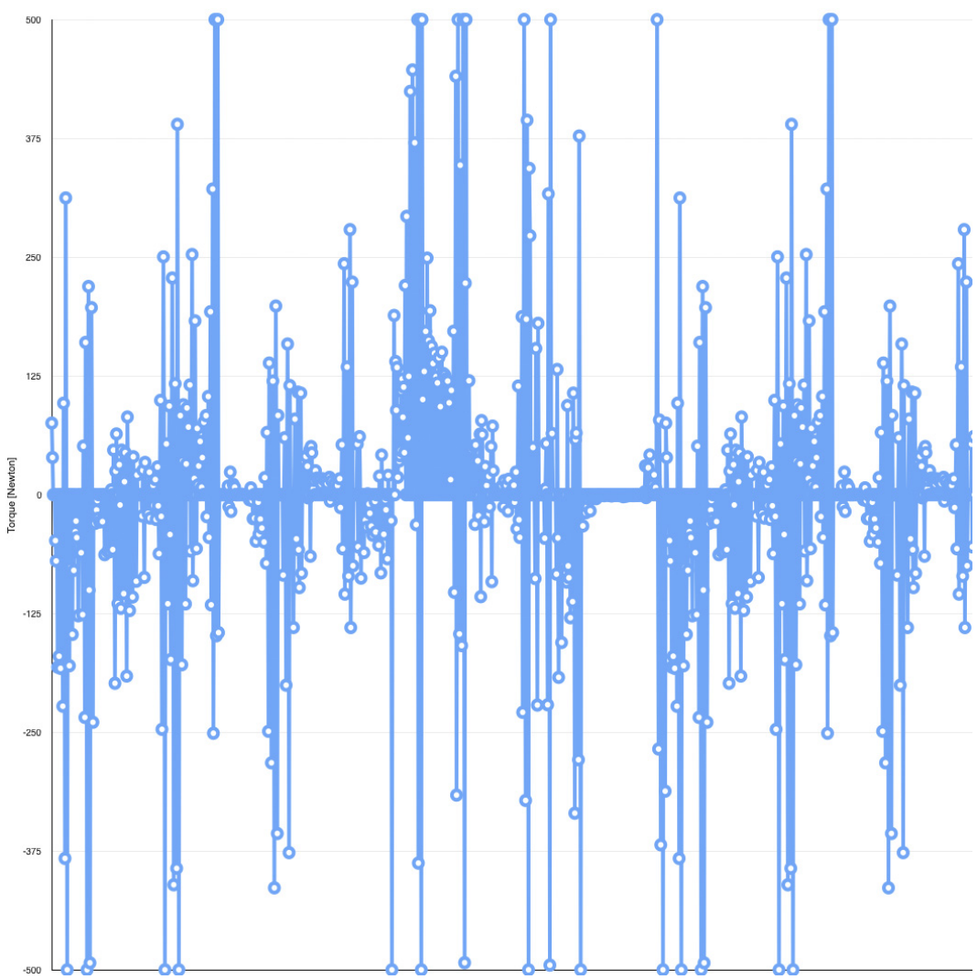


Figure 7.116: Uniform Movement Base Torque, kp = 250

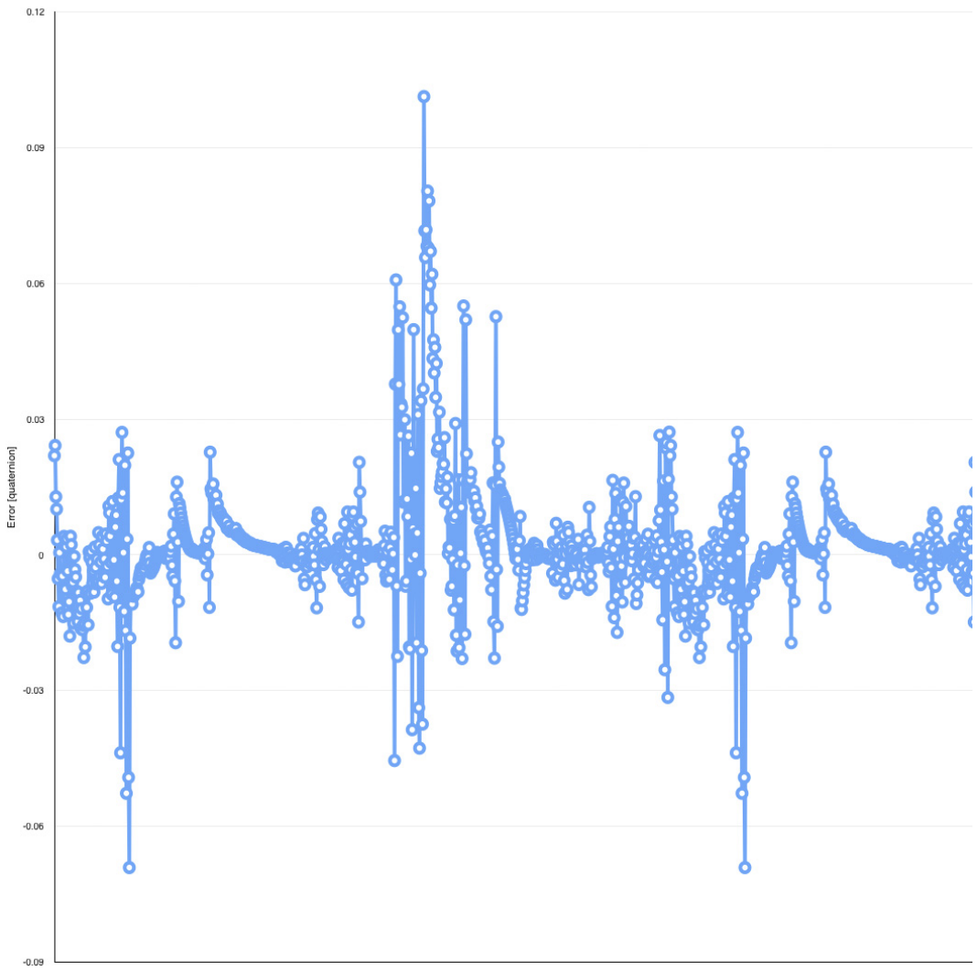


Figure 7.117: Uniform Movement Base Position error, kp = 300

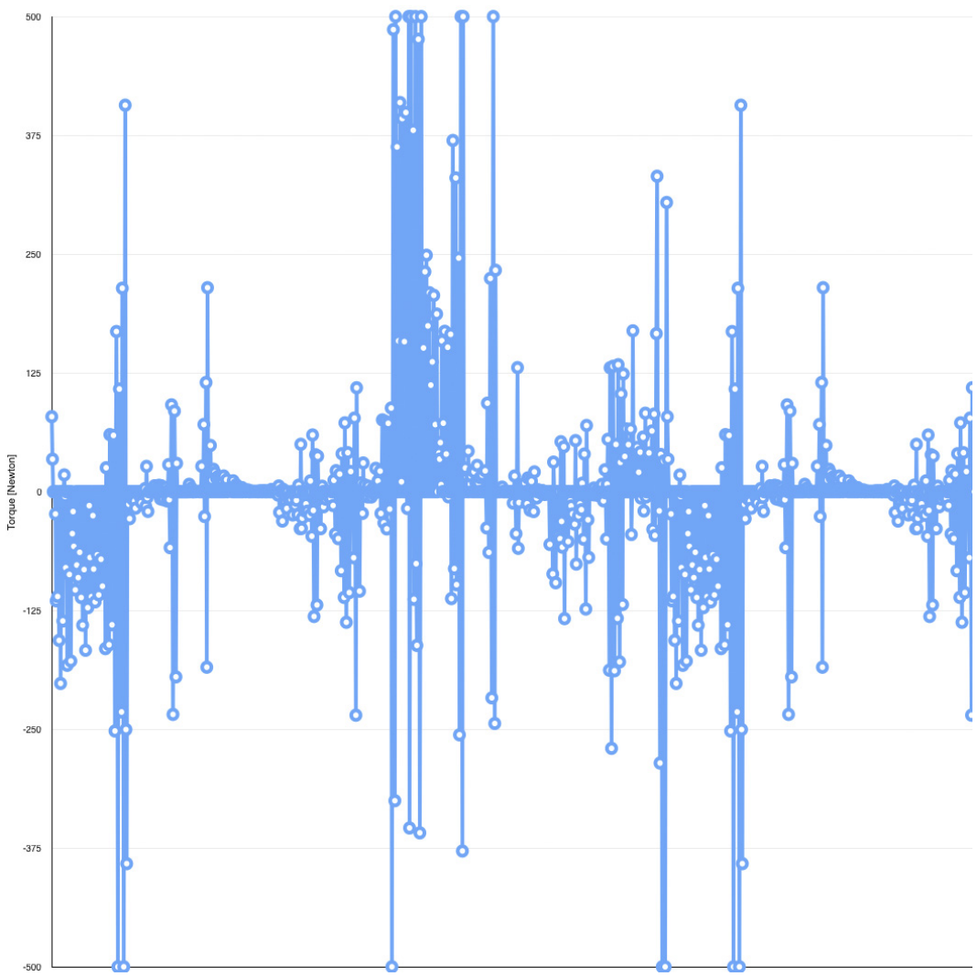


Figure 7.118: Uniform Movement Base Torque, kp = 300

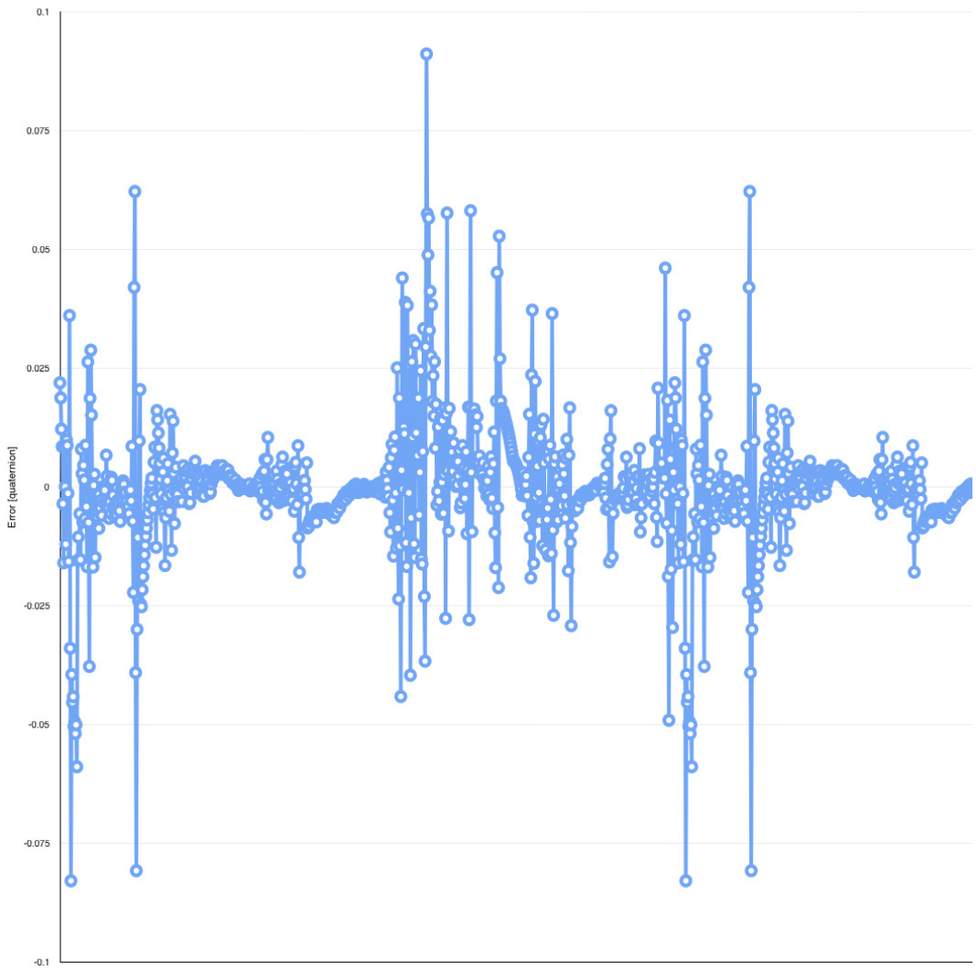


Figure 7.119: Uniform Movement Base Position error, kp = 350

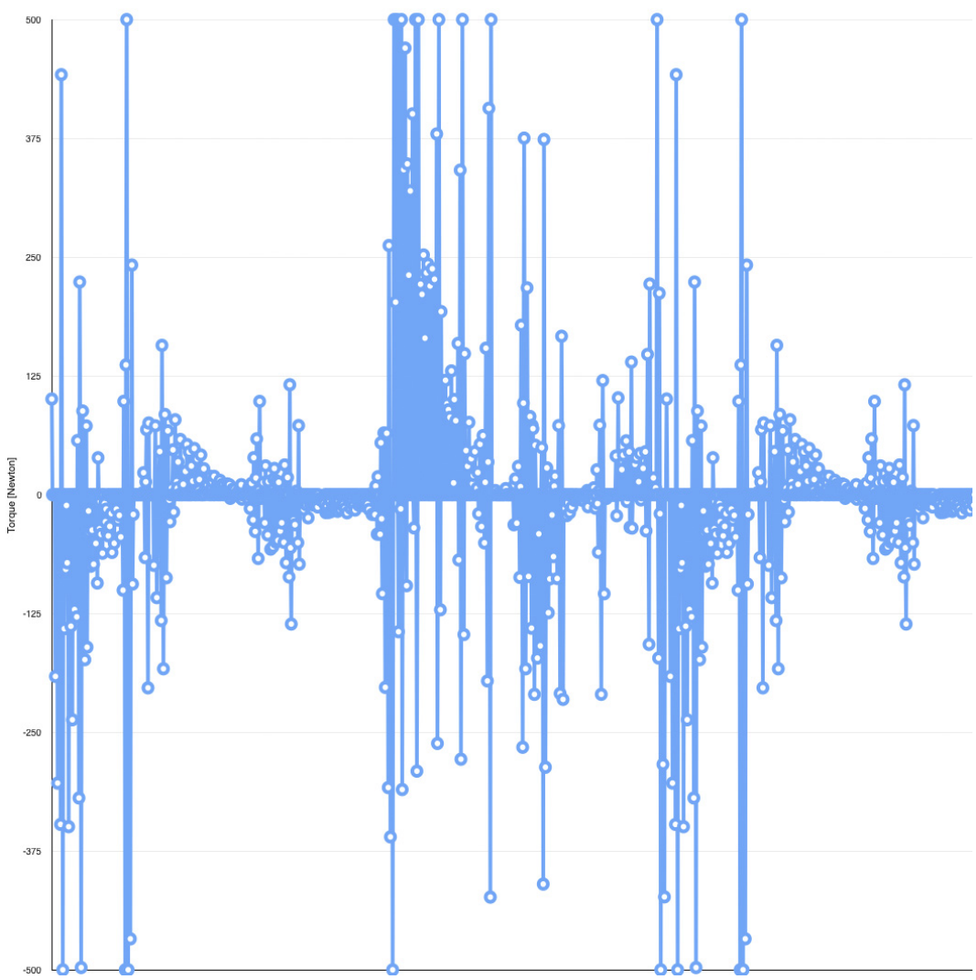


Figure 7.120: Uniform Movement Base Torque, kp = 350

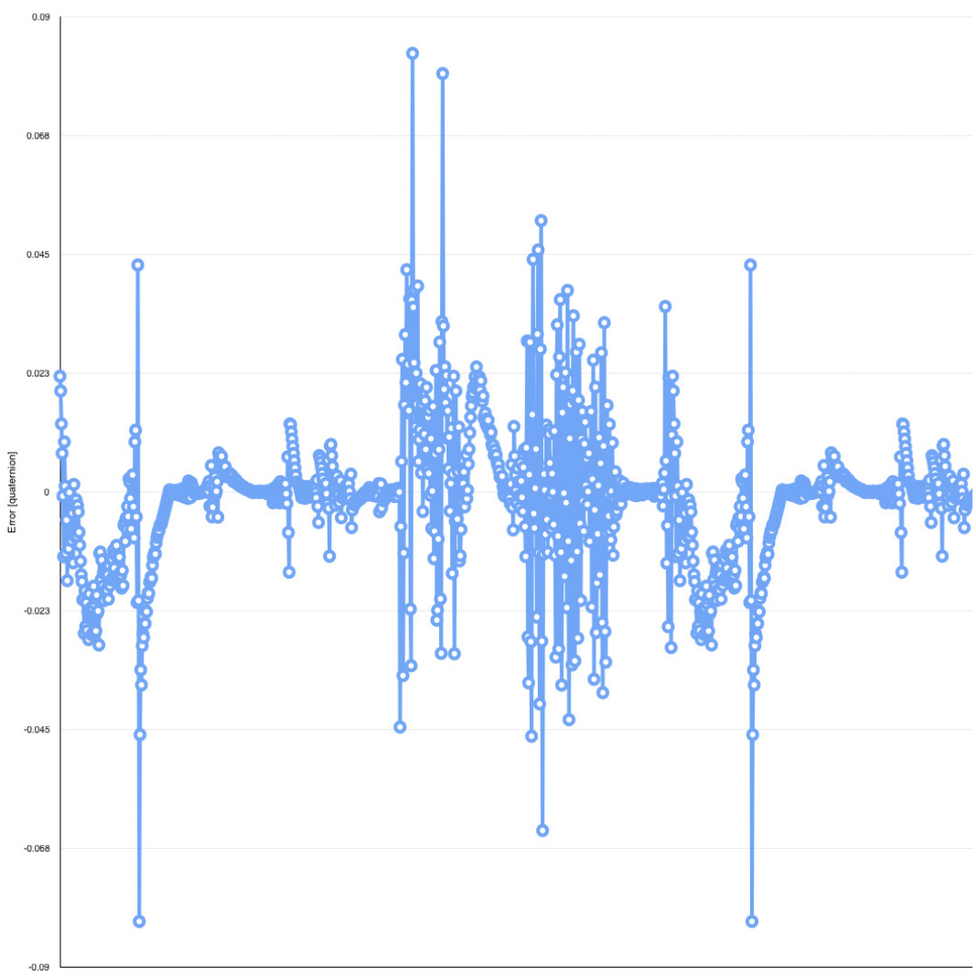


Figure 7.121: Uniform Movement Base Position error, kp = 400

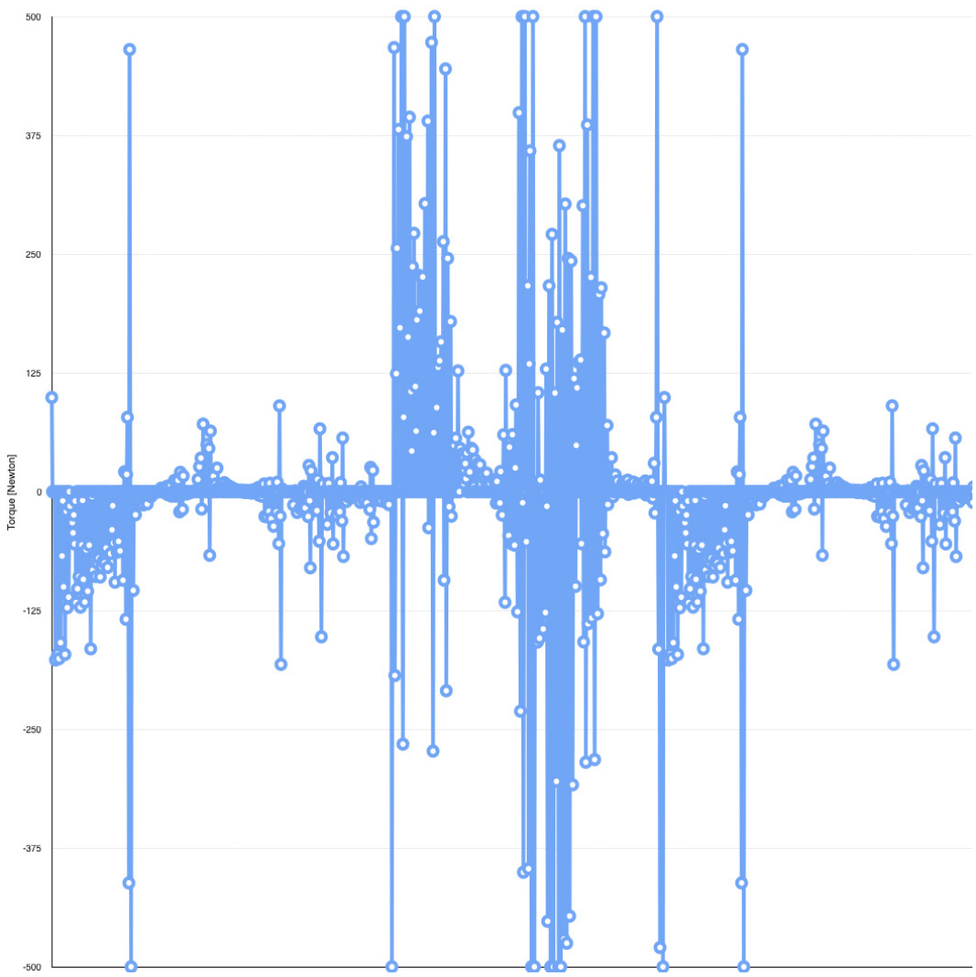


Figure 7.122: Uniform Movement Base Torque, kp = 400

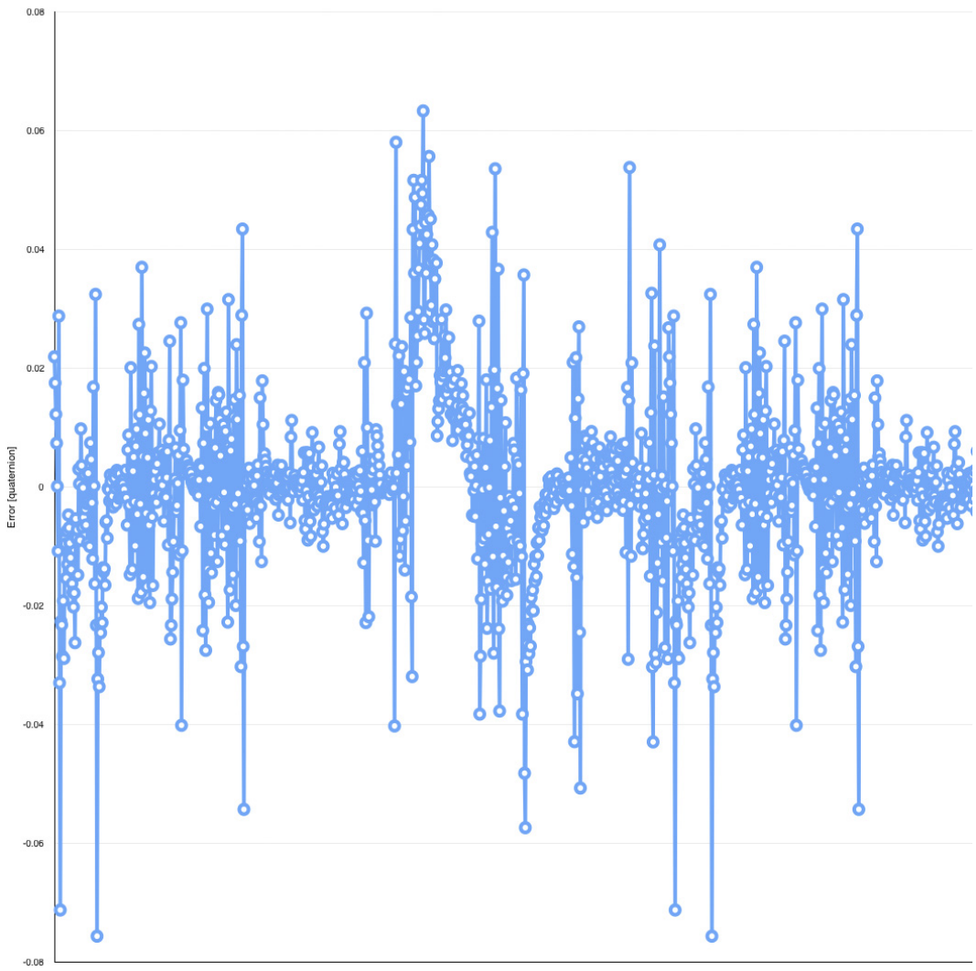


Figure 7.123: Uniform Movement Base Position error, kp = 500

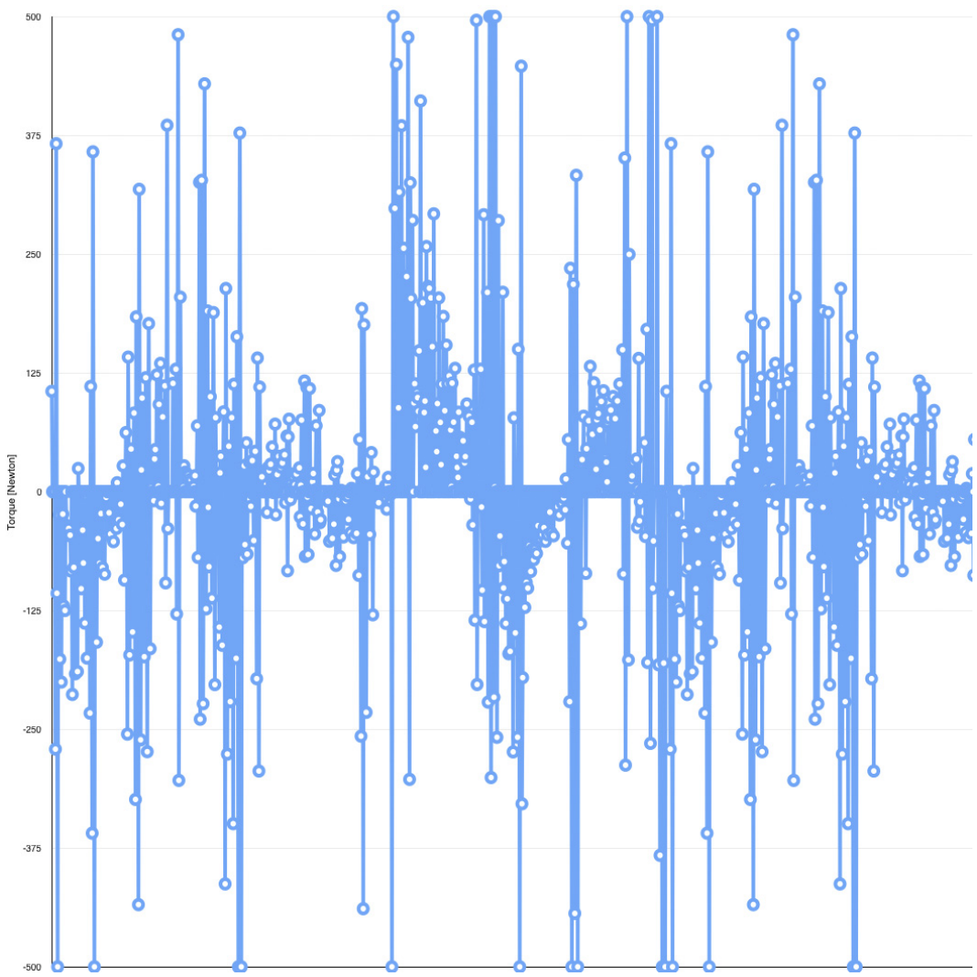


Figure 7.124: Uniform Movement Base Torque, kp = 500

7.1.2 Integral variable

7.1.2.1 Joint 1

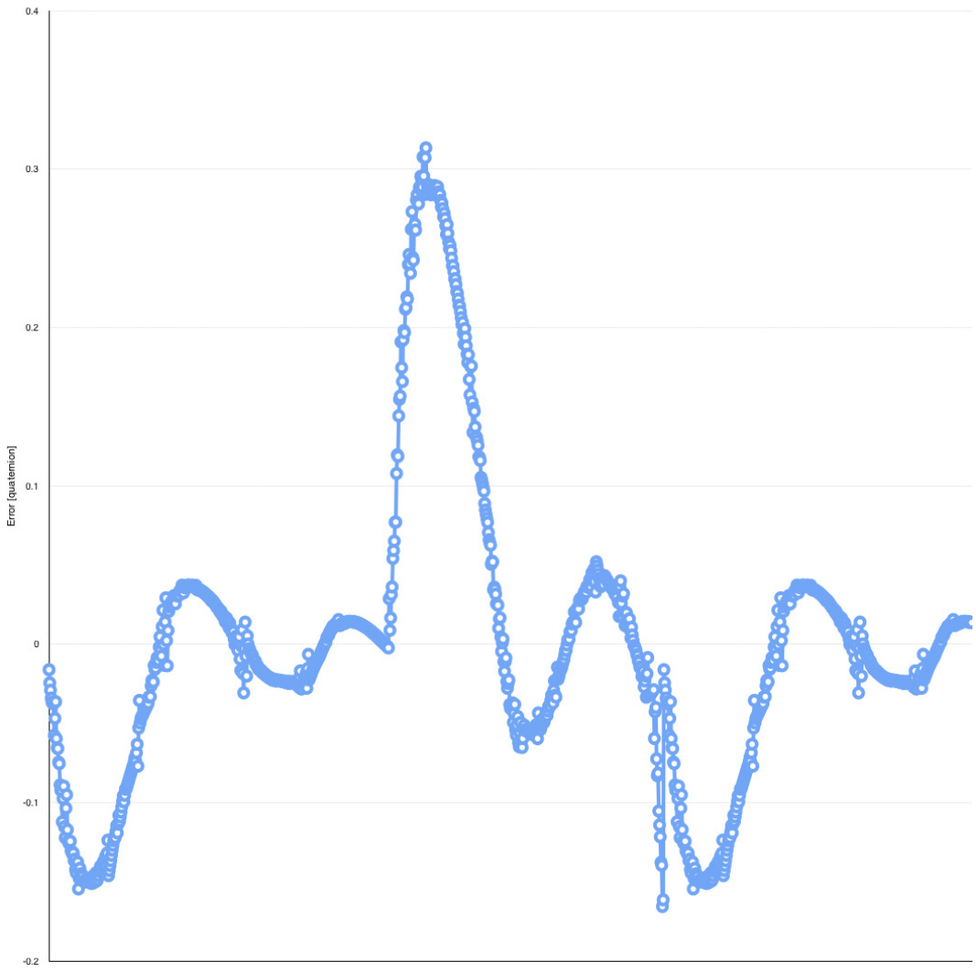


Figure 7.125: Uniform Movement Base Position error, Ki = 100

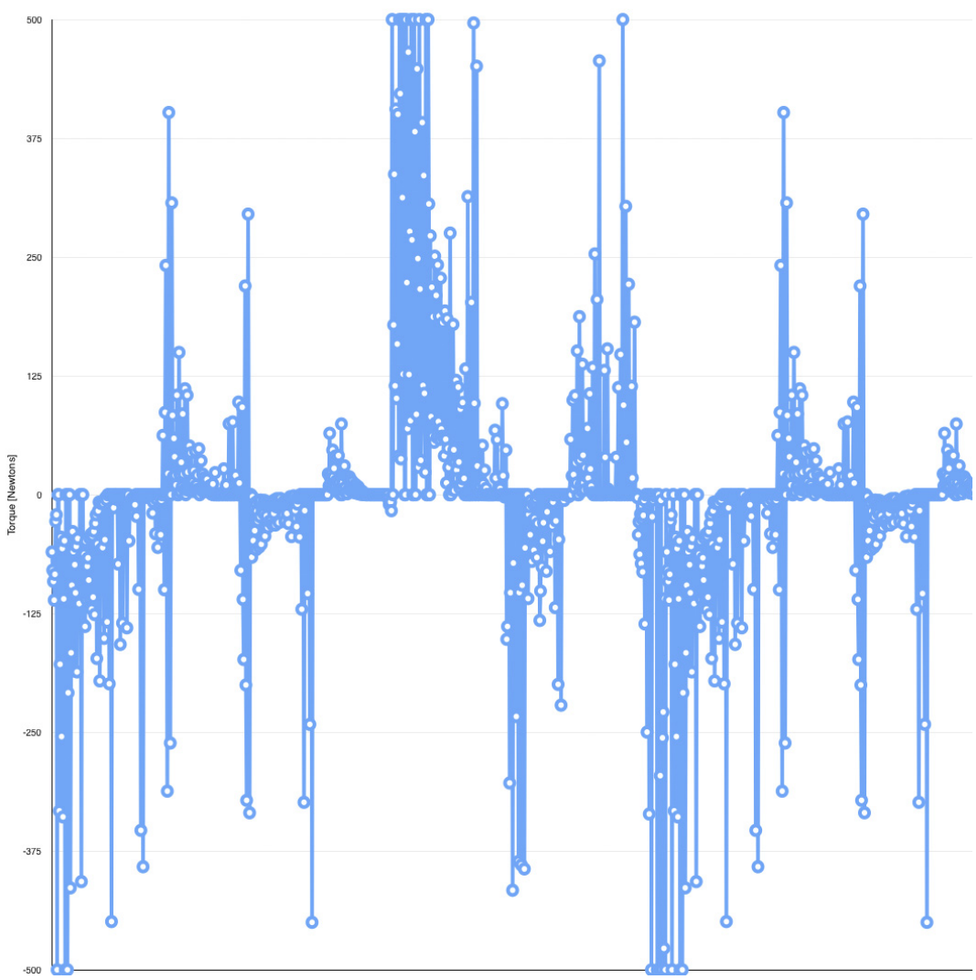


Figure 7.126: Uniform Movement Base Torque, Ki = 100

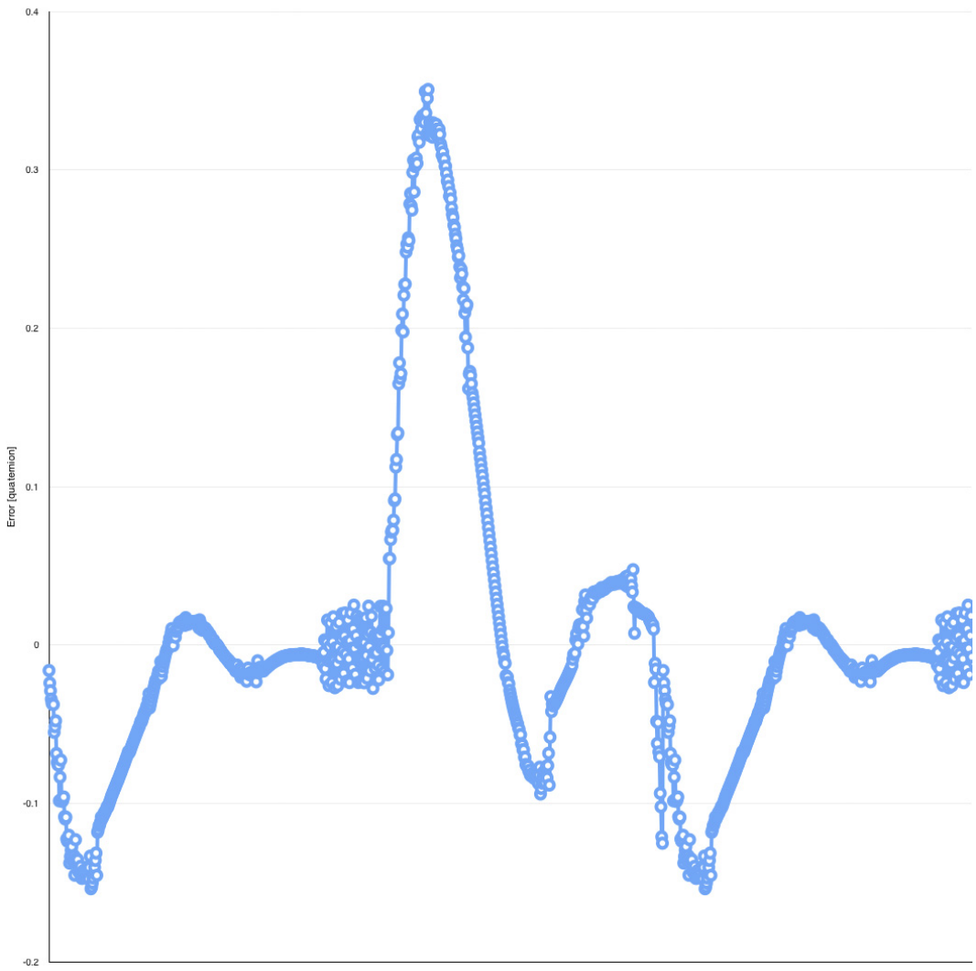


Figure 7.127: Uniform Movement Base Position error, Ki = 150

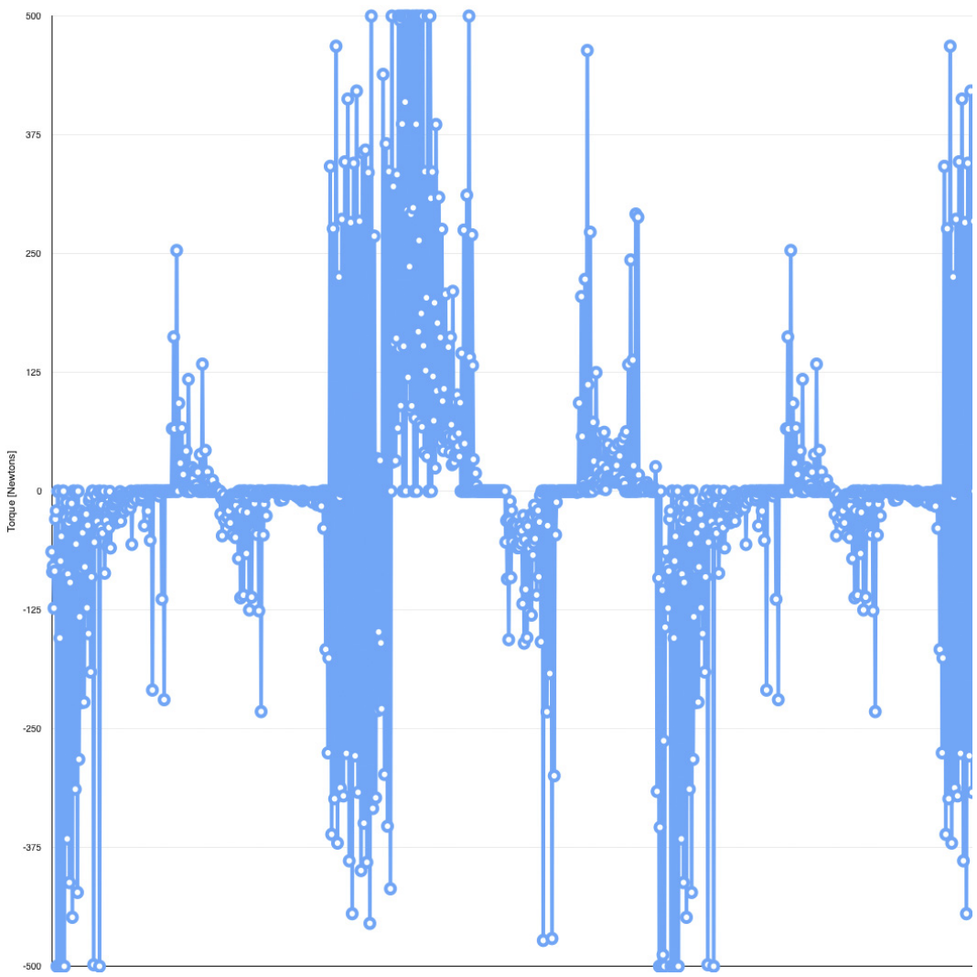


Figure 7.128: Uniform Movement Base Torque, Ki = 150

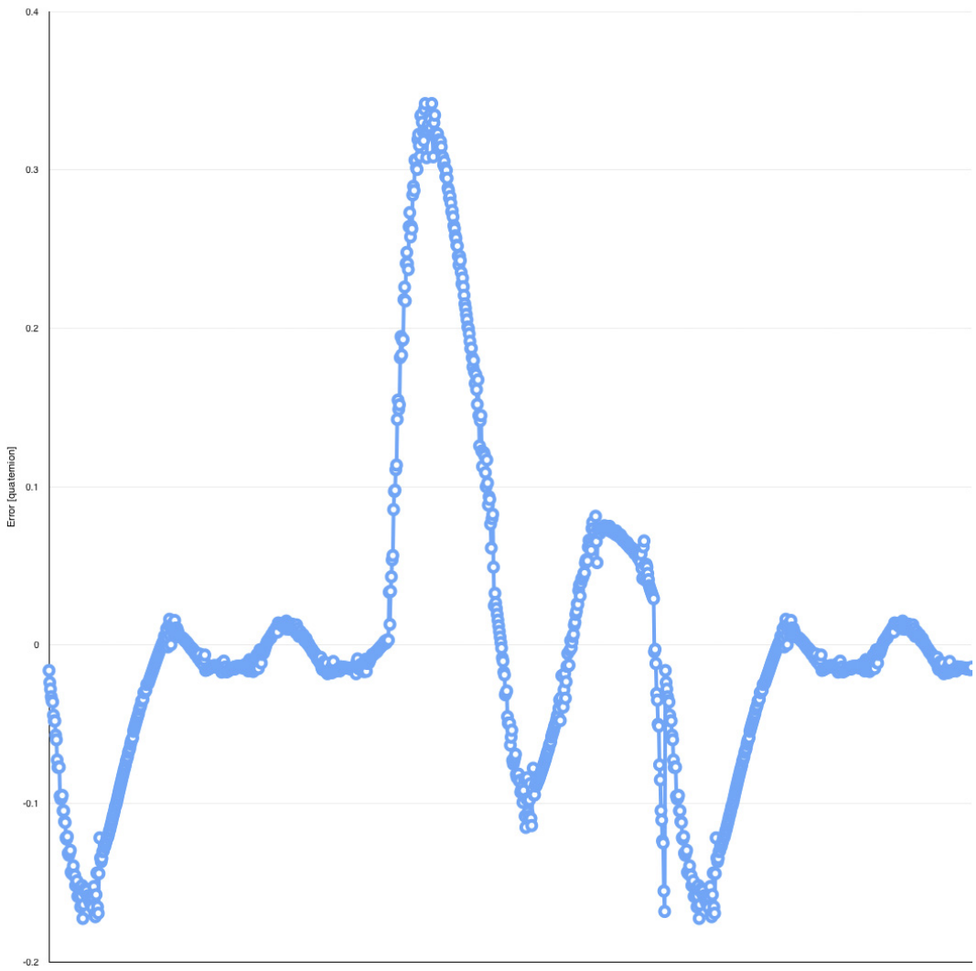


Figure 7.129: Uniform Movement Base Position error, Ki = 200

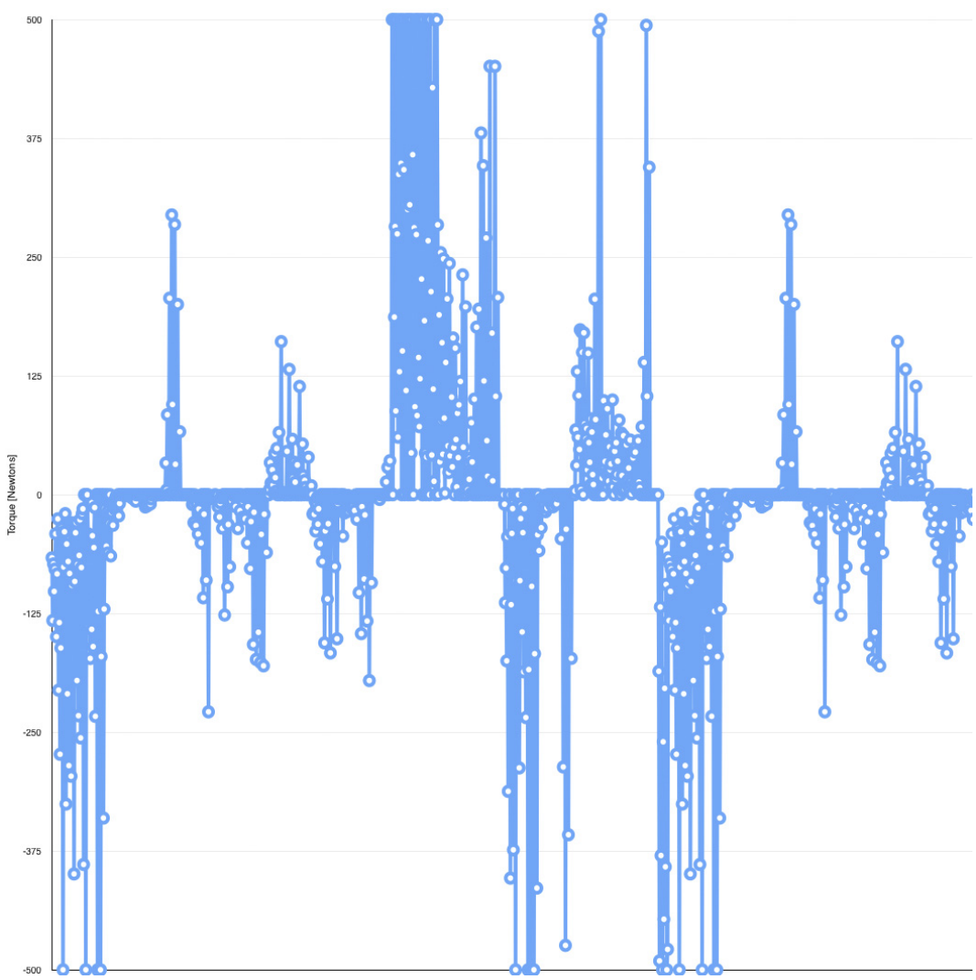


Figure 7.130: Uniform Movement Base Torque, Ki = 200

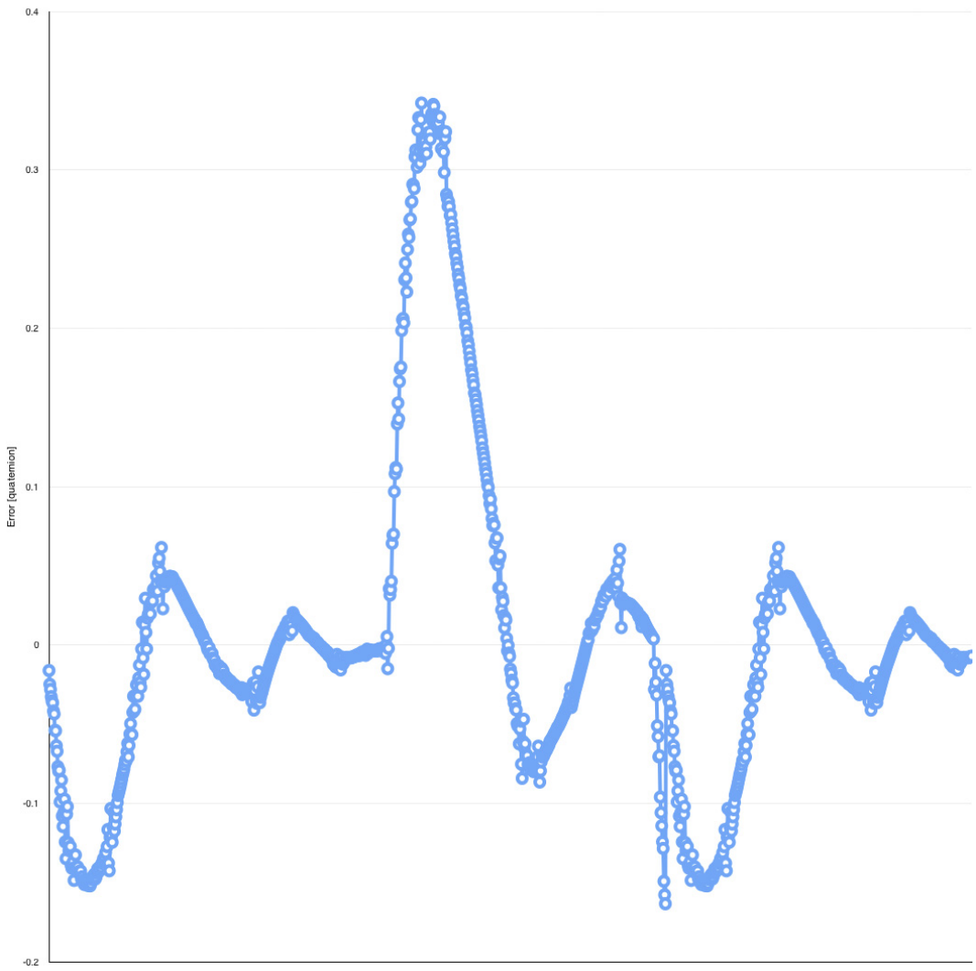


Figure 7.131: Uniform Movement Base Position error, Ki = 250

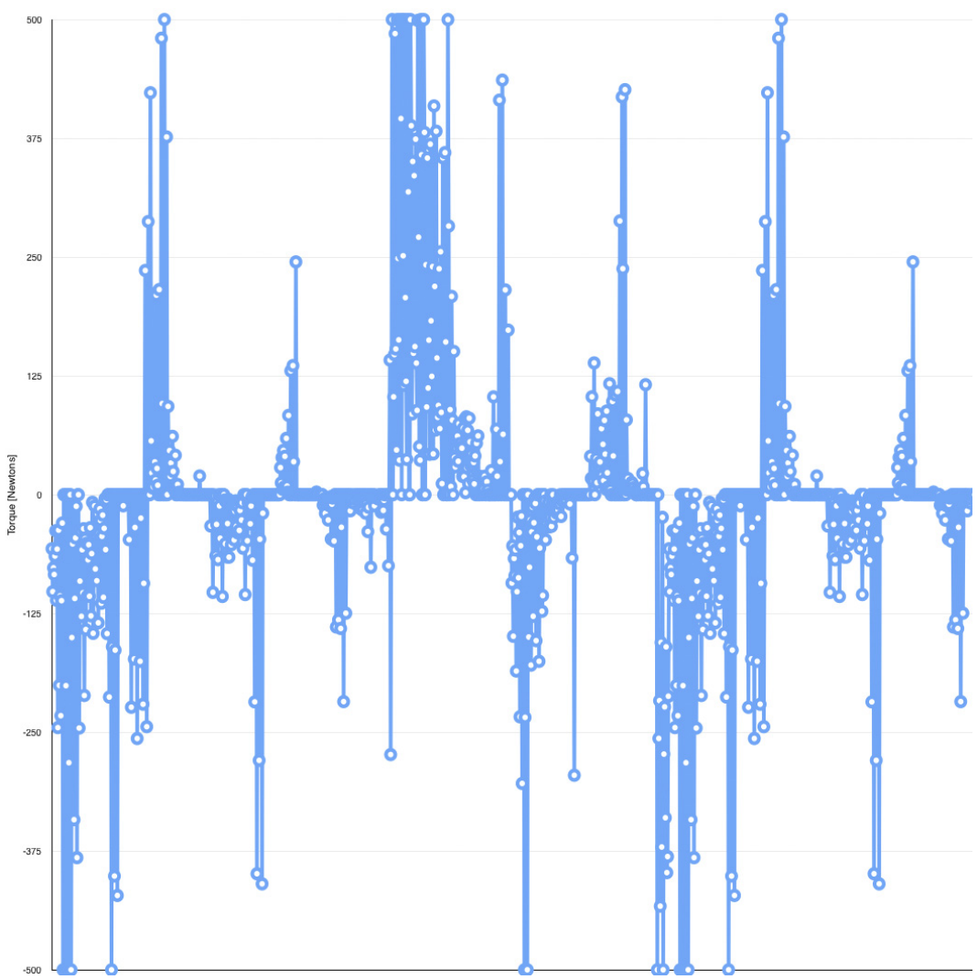


Figure 7.132: Uniform Movement Base Torque, Ki = 250

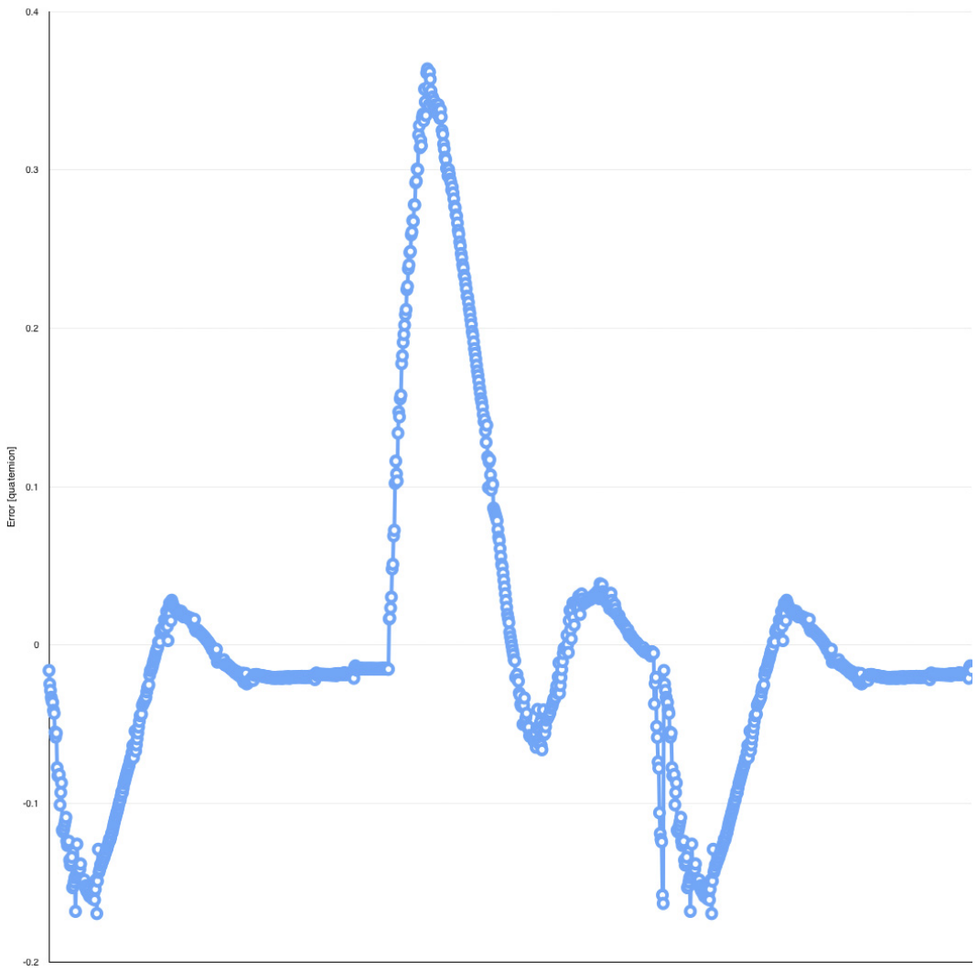


Figure 7.133: Uniform Movement Base Position error, Ki = 300

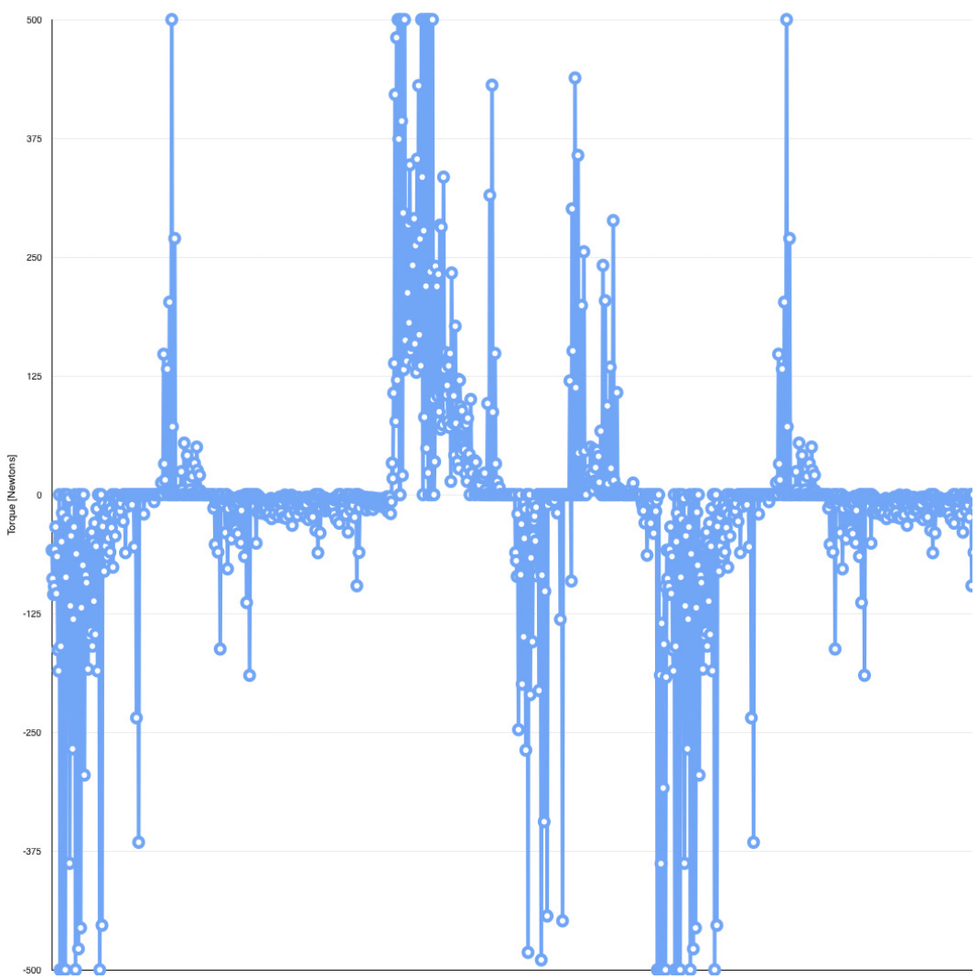


Figure 7.134: Uniform Movement Base Torque, Ki = 300

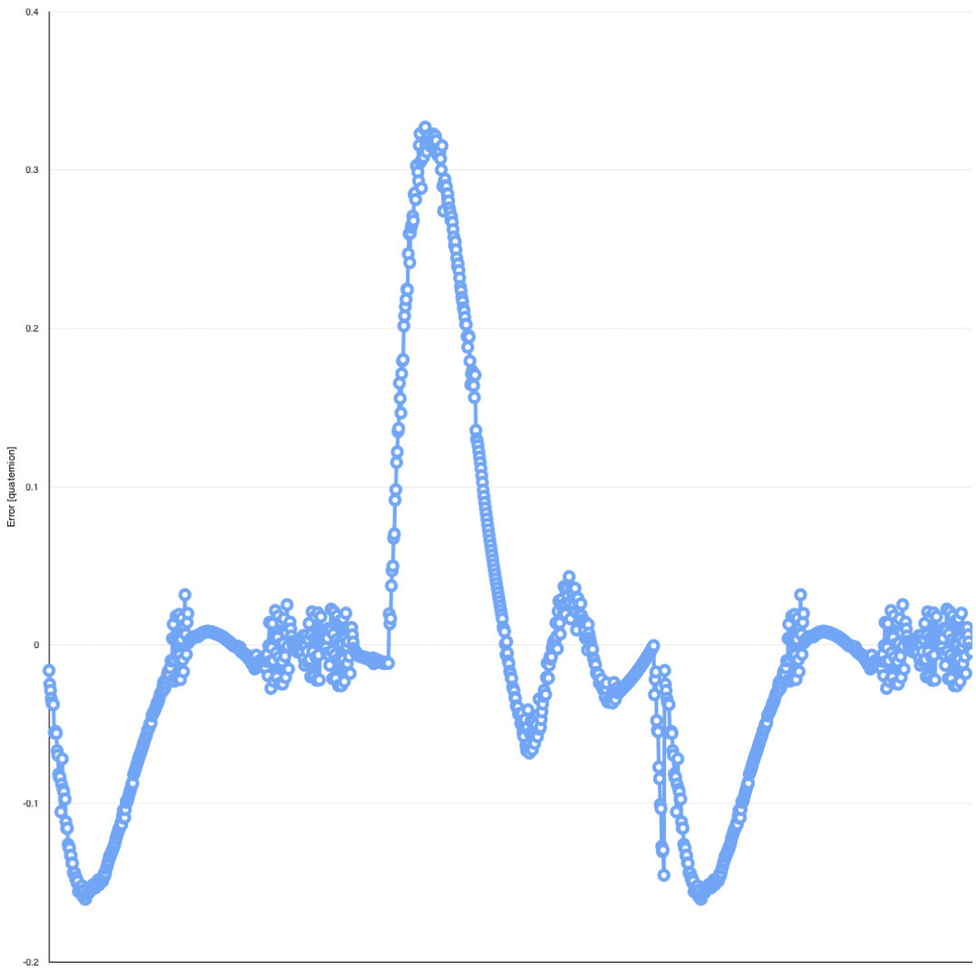


Figure 7.135: Uniform Movement Base Position error, Ki = 350

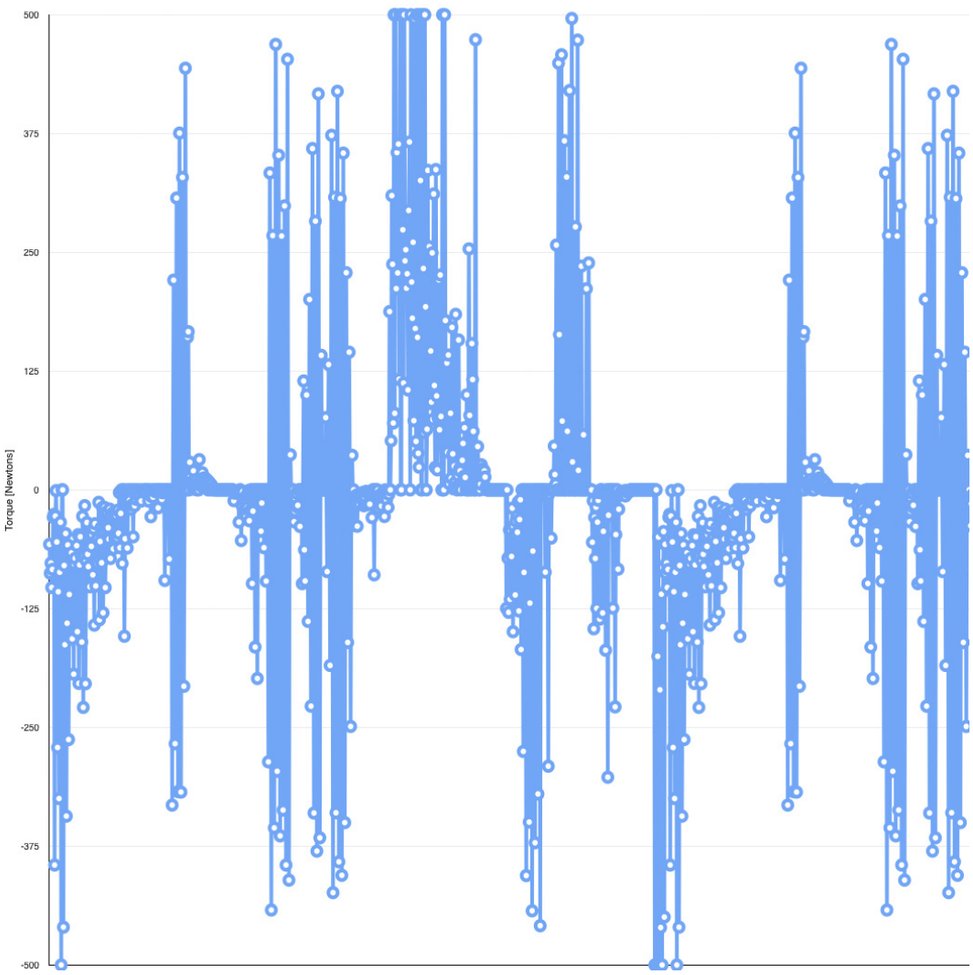


Figure 7.136: Uniform Movement Base Torque, Ki = 350

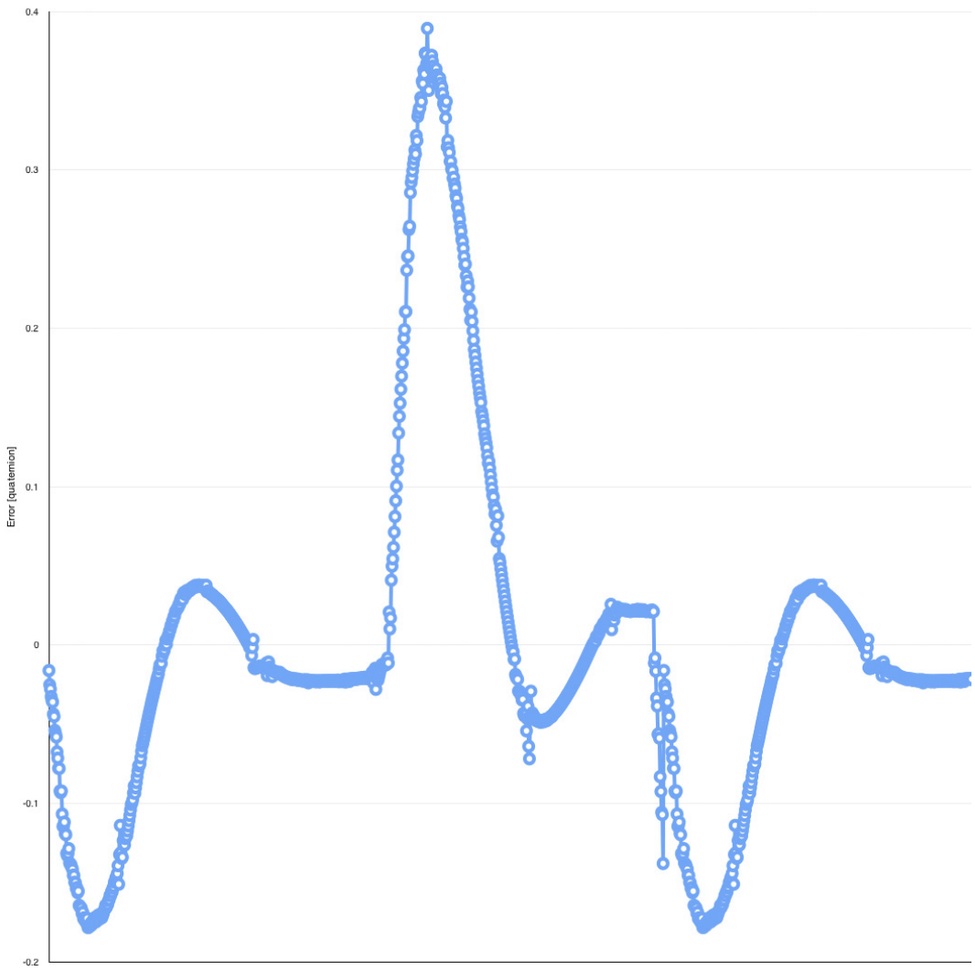


Figure 7.137: Uniform Movement Base Position error, Ki = 400

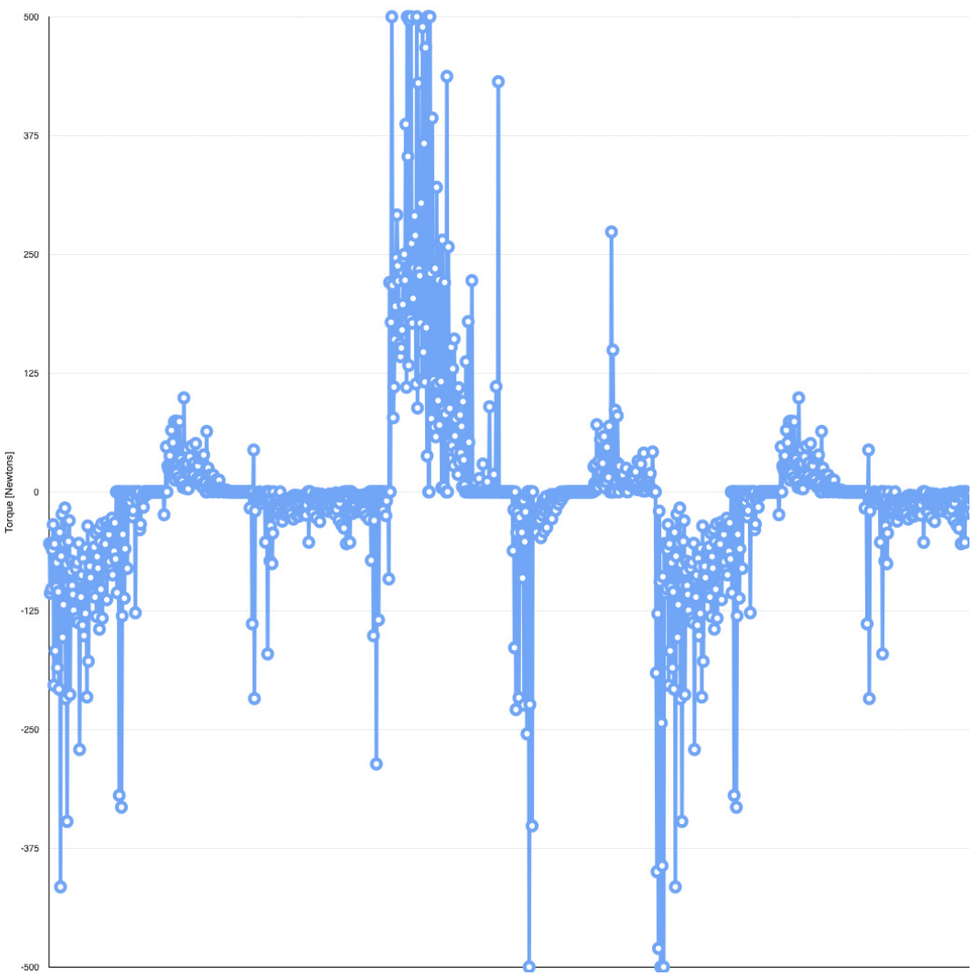


Figure 7.138: Uniform Movement Base Torque, Ki = 400

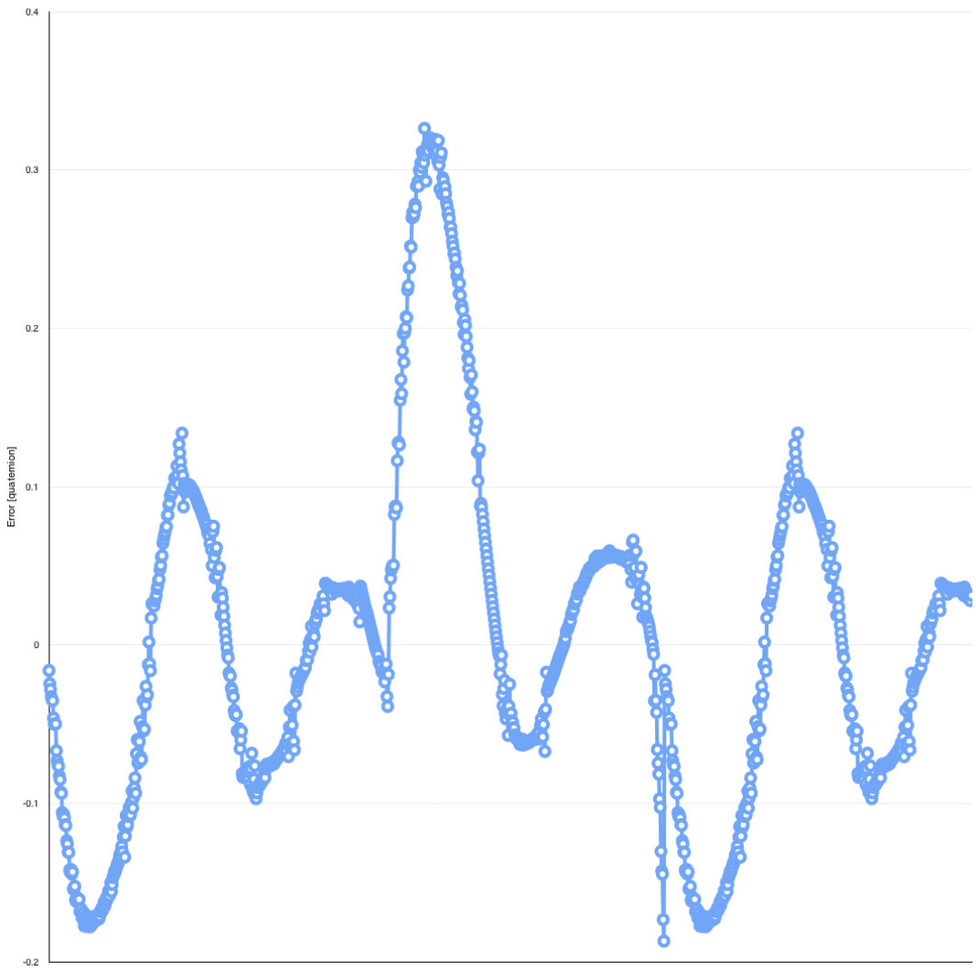


Figure 7.139: Uniform Movement Base Position error, Ki = 450

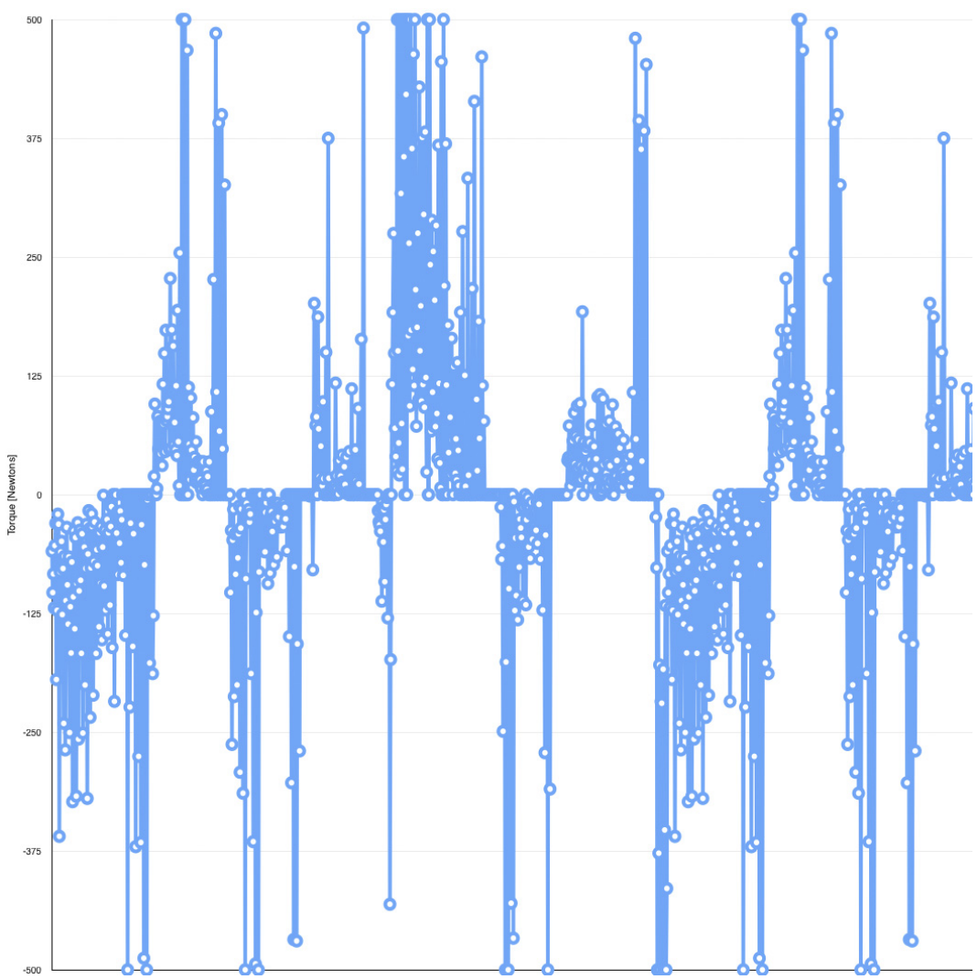


Figure 7.140: Uniform Movement Base Torque, Ki = 450

7.1.2.2 Joint 2

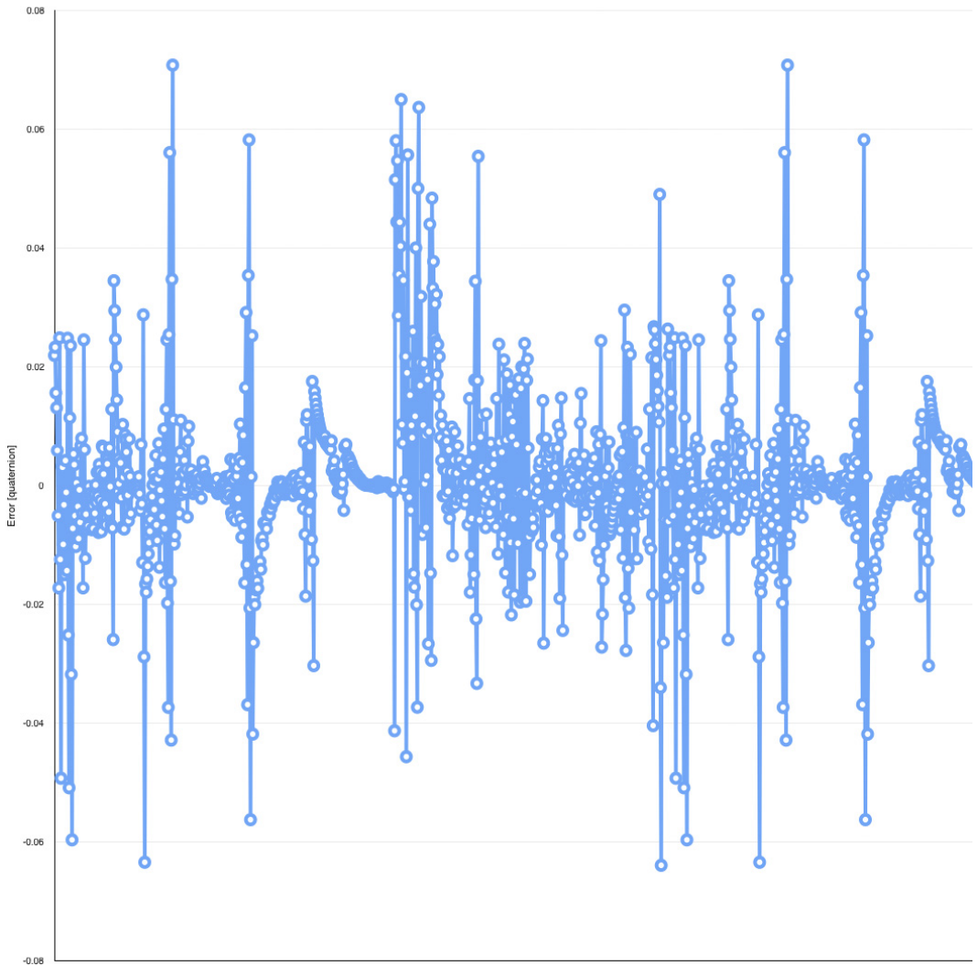


Figure 7.141: Uniform Movement Base Position error, Ki = 100

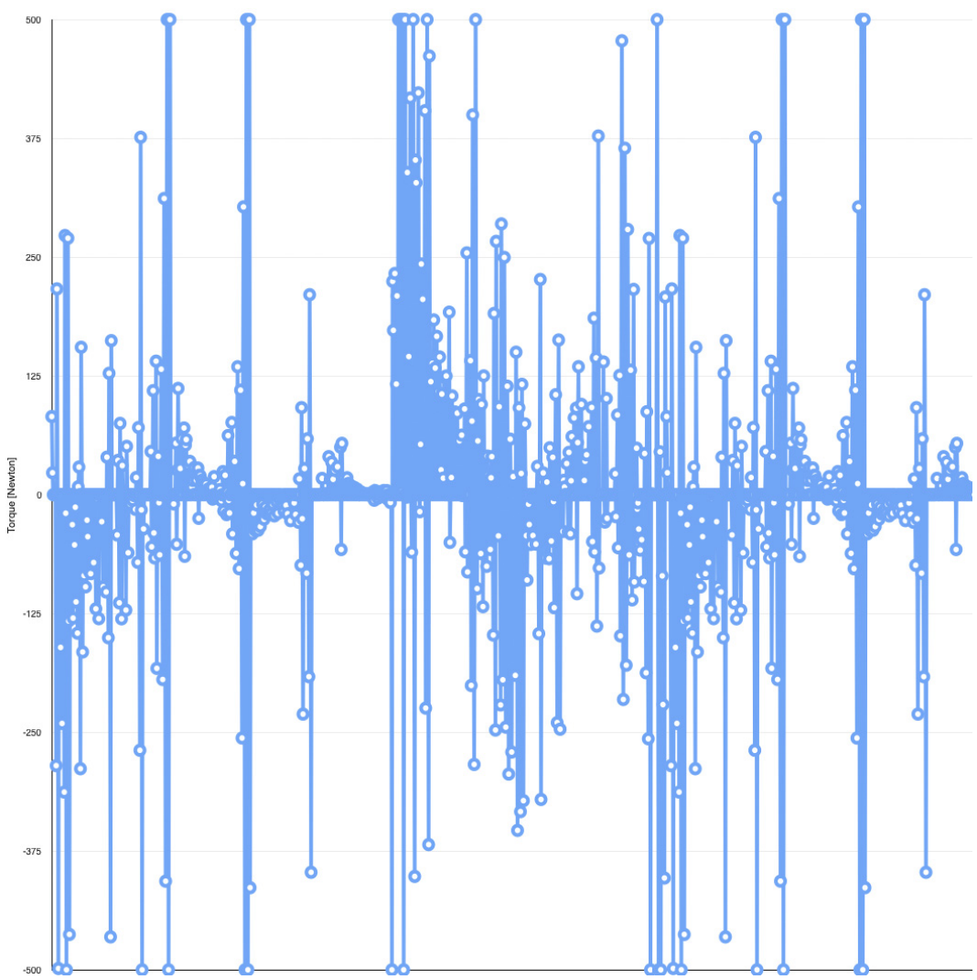


Figure 7.142: Uniform Movement Base Torque, Ki = 100

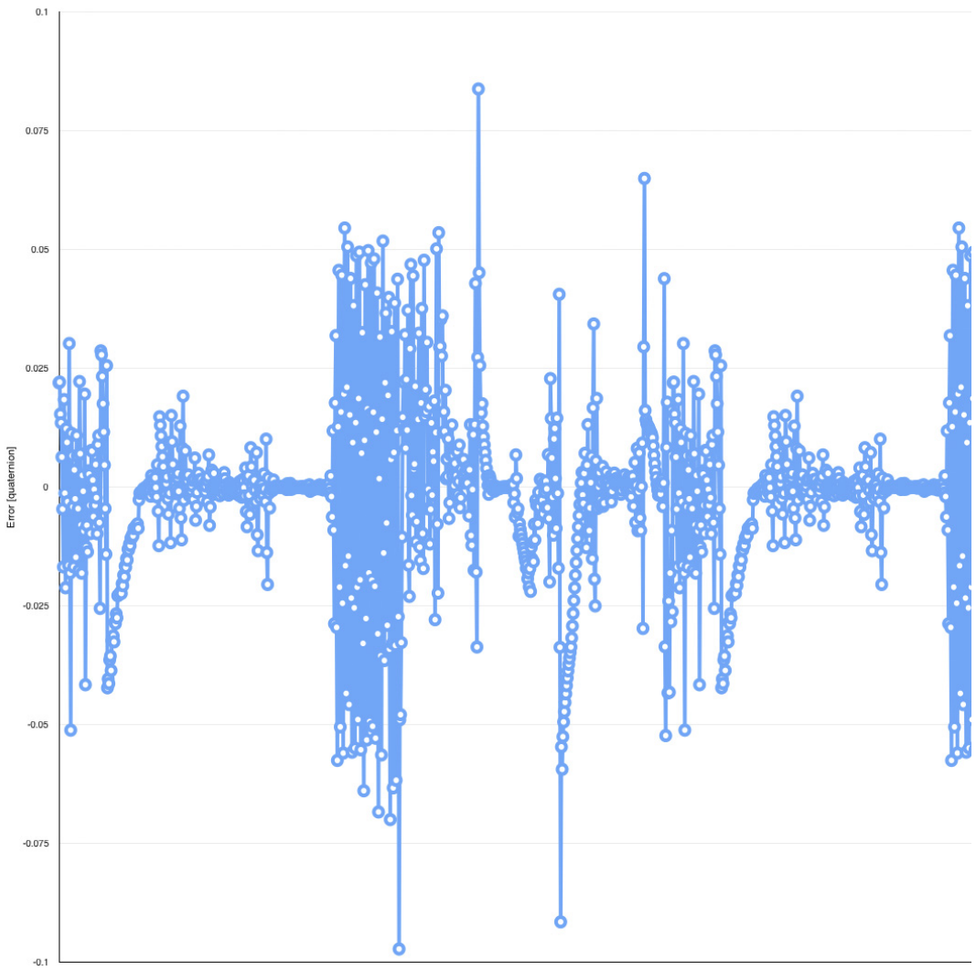


Figure 7.143: Uniform Movement Base Position error, Ki = 150

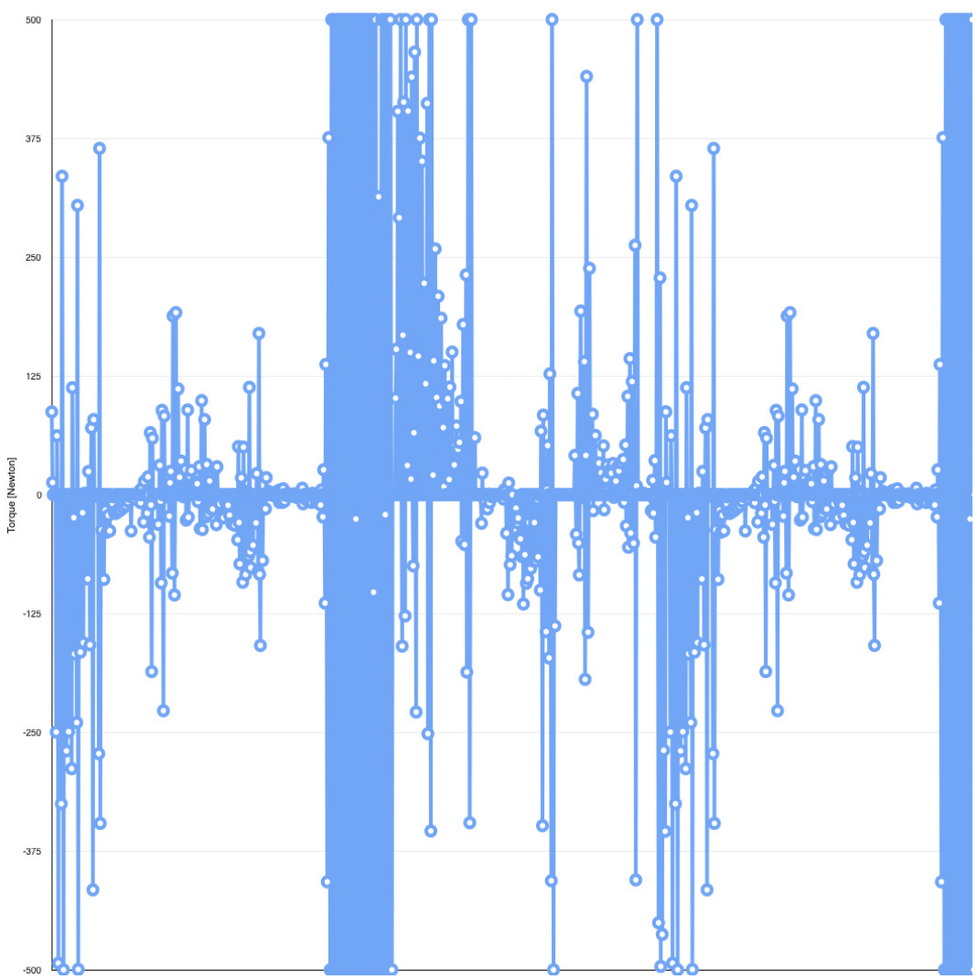


Figure 7.144: Uniform Movement Base Torque, Ki = 150

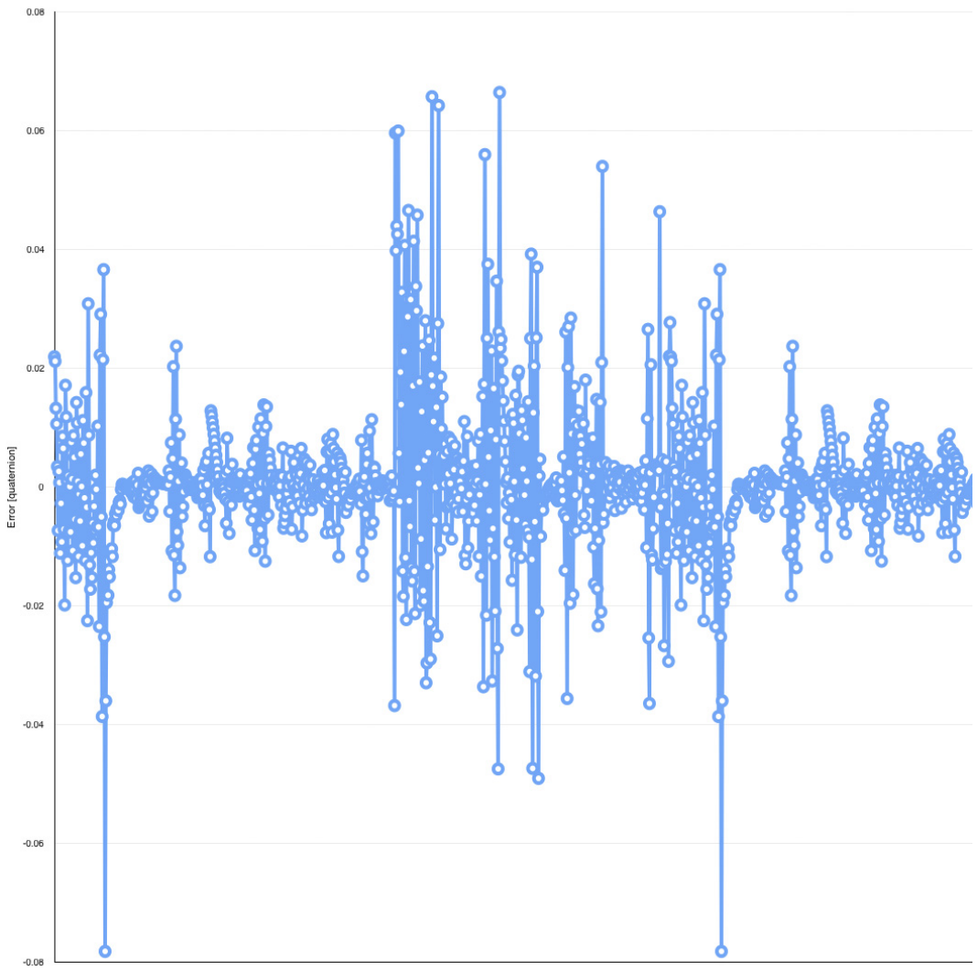


Figure 7.145: Uniform Movement Base Position error, Ki = 200

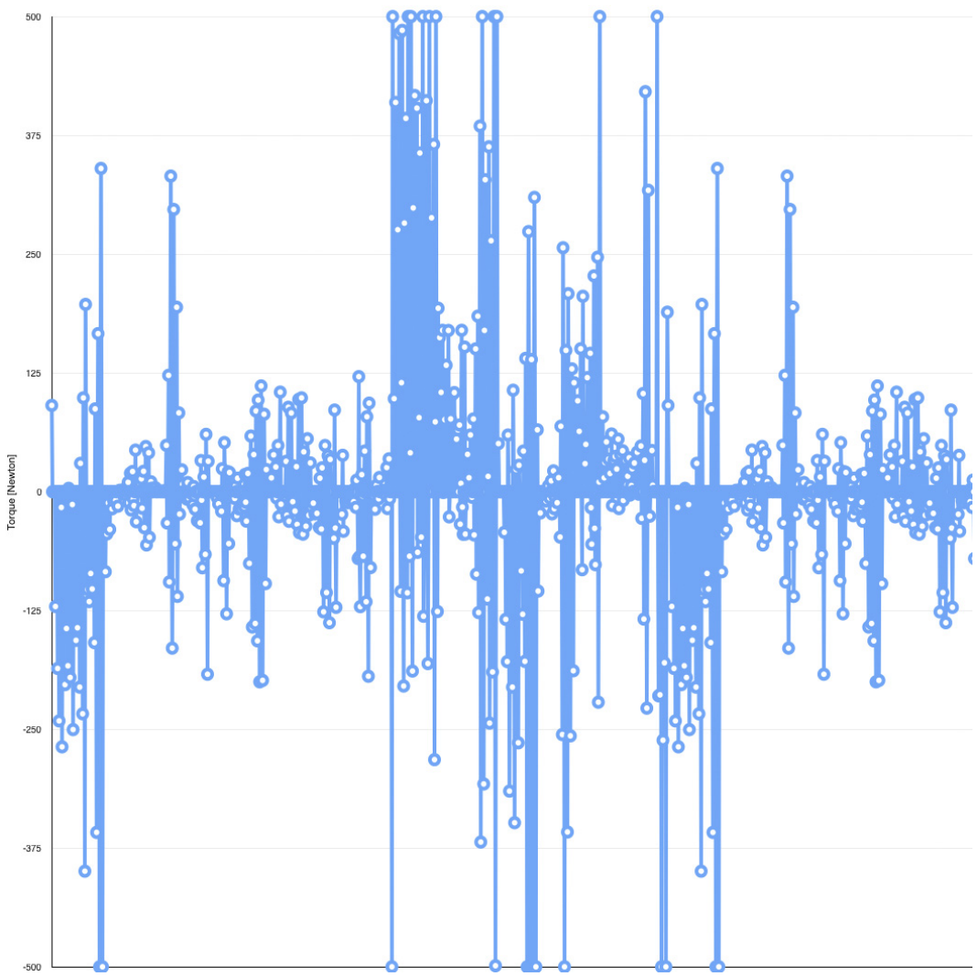


Figure 7.146: Uniform Movement Base Torque, Ki = 200

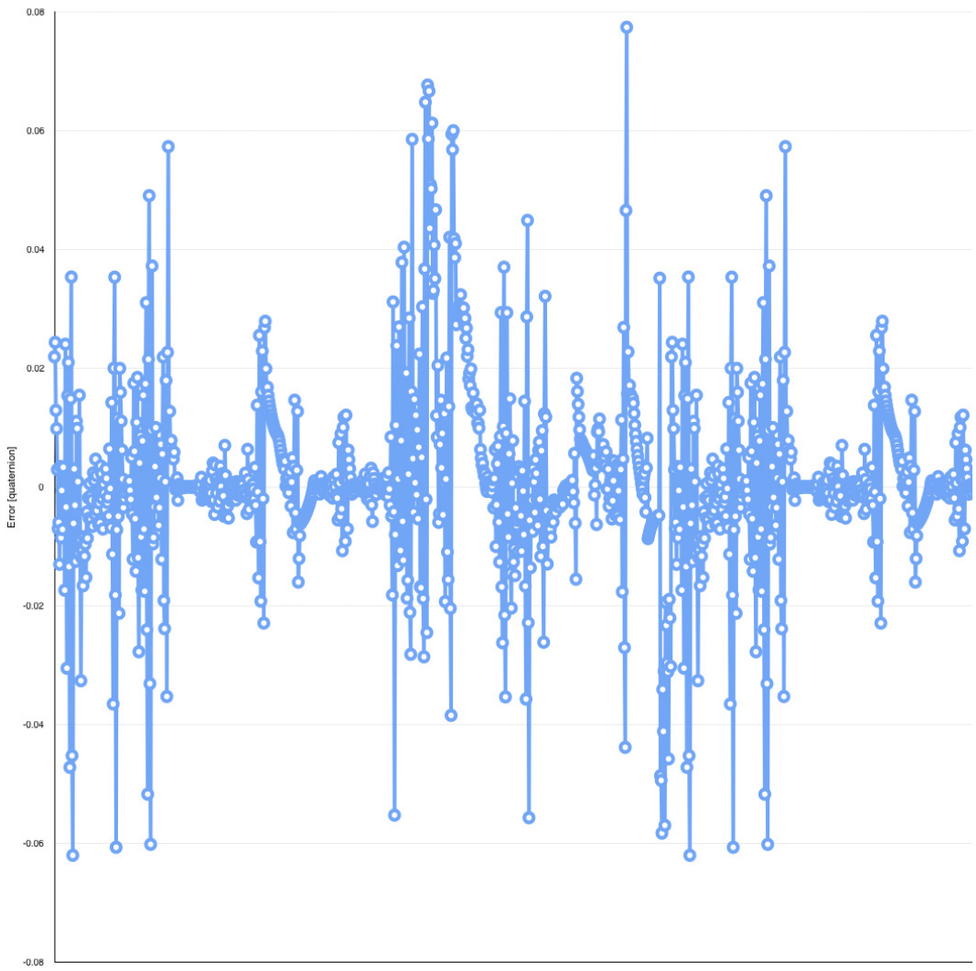


Figure 7.147: Uniform Movement Base Position error, Ki = 250

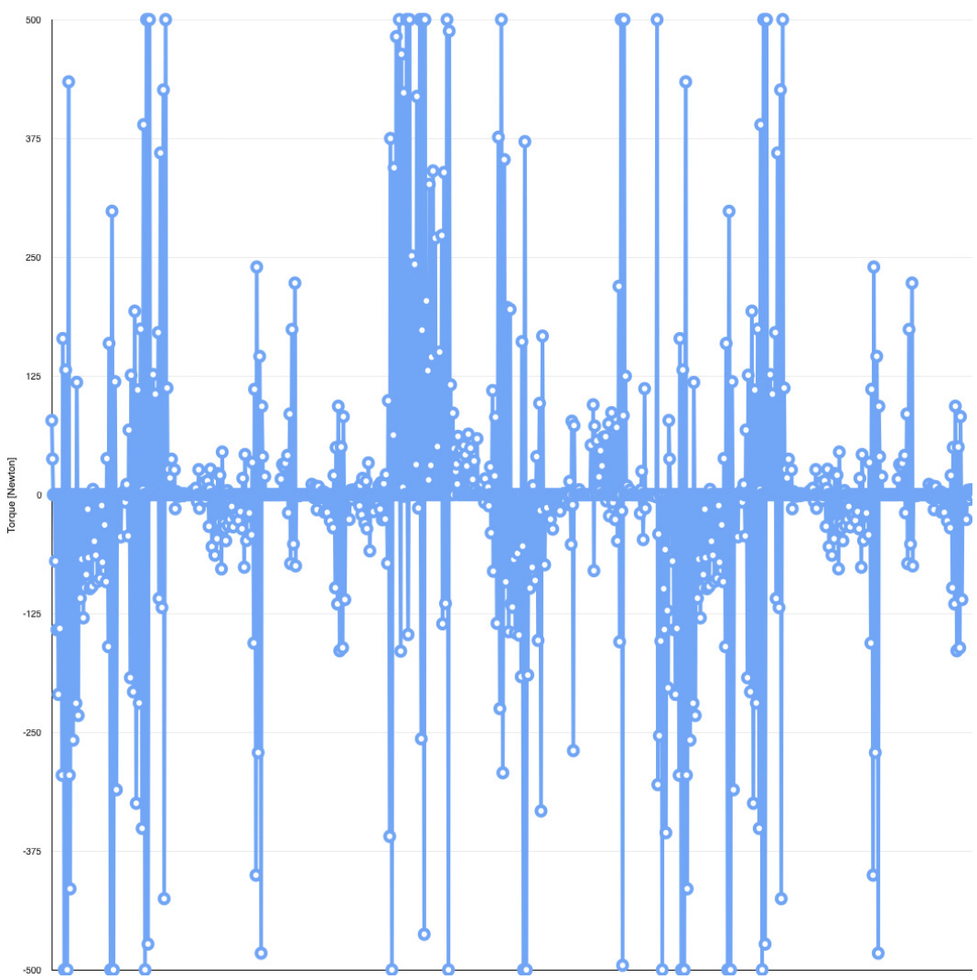


Figure 7.148: Uniform Movement Base Torque, Ki = 250

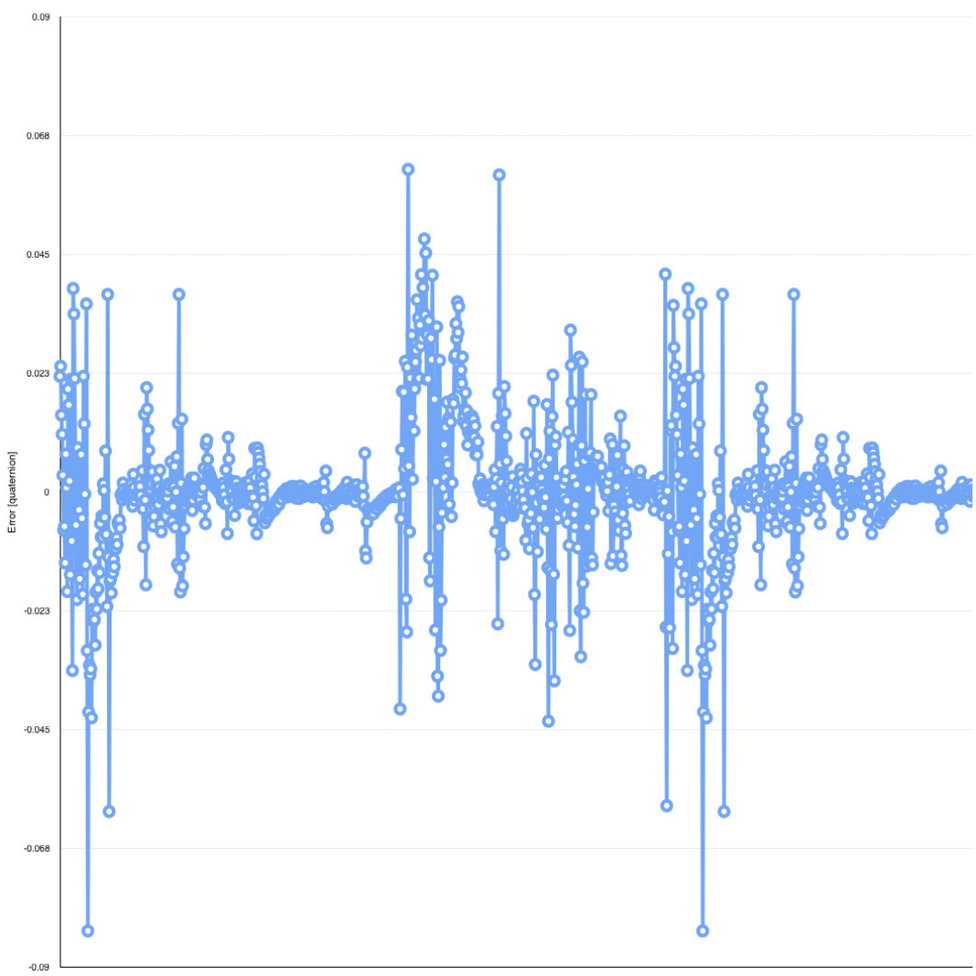


Figure 7.149: Uniform Movement Base Position error, Ki = 300

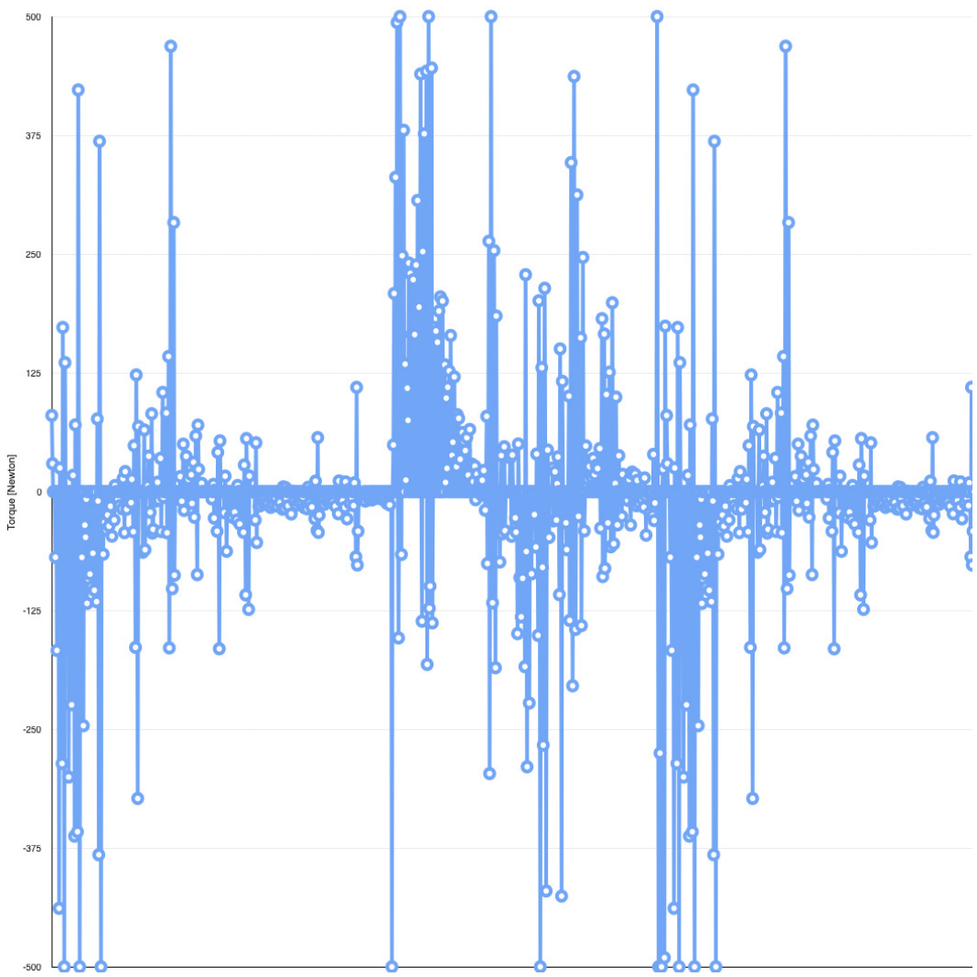


Figure 7.150: Uniform Movement Base Torque, Ki = 300

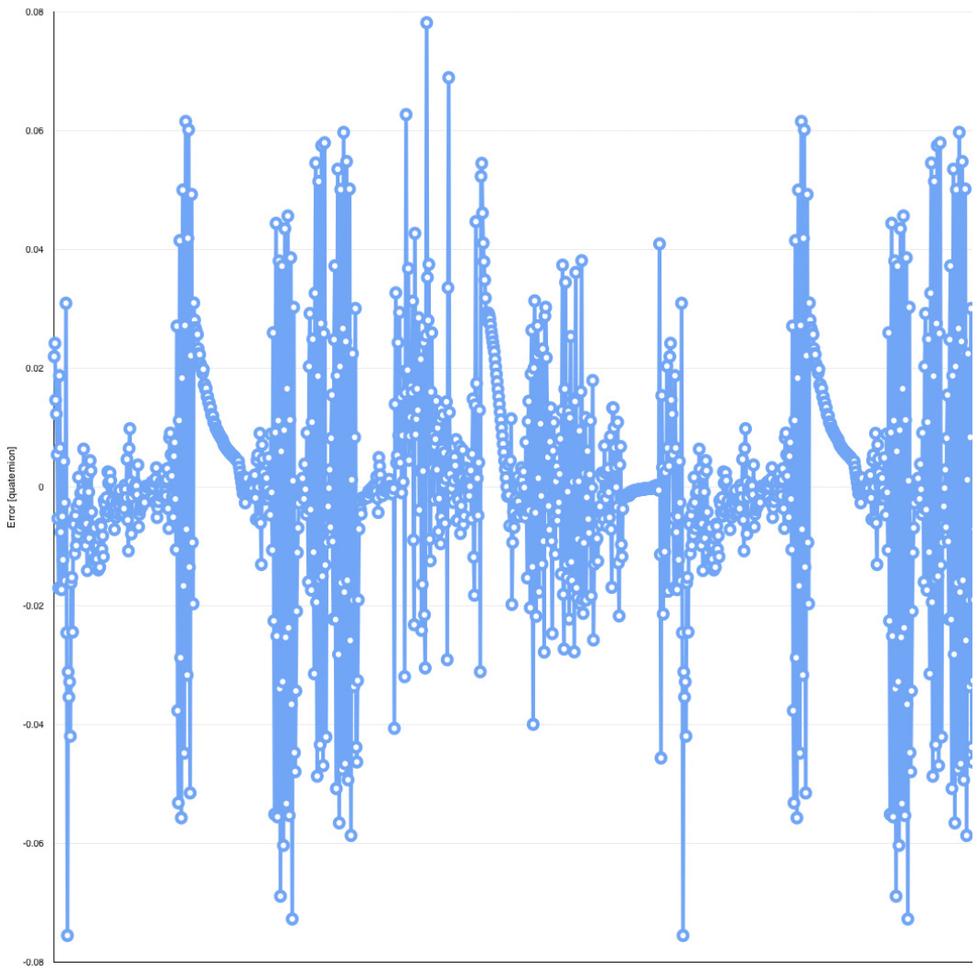


Figure 7.151: Uniform Movement Base Position error, Ki = 350

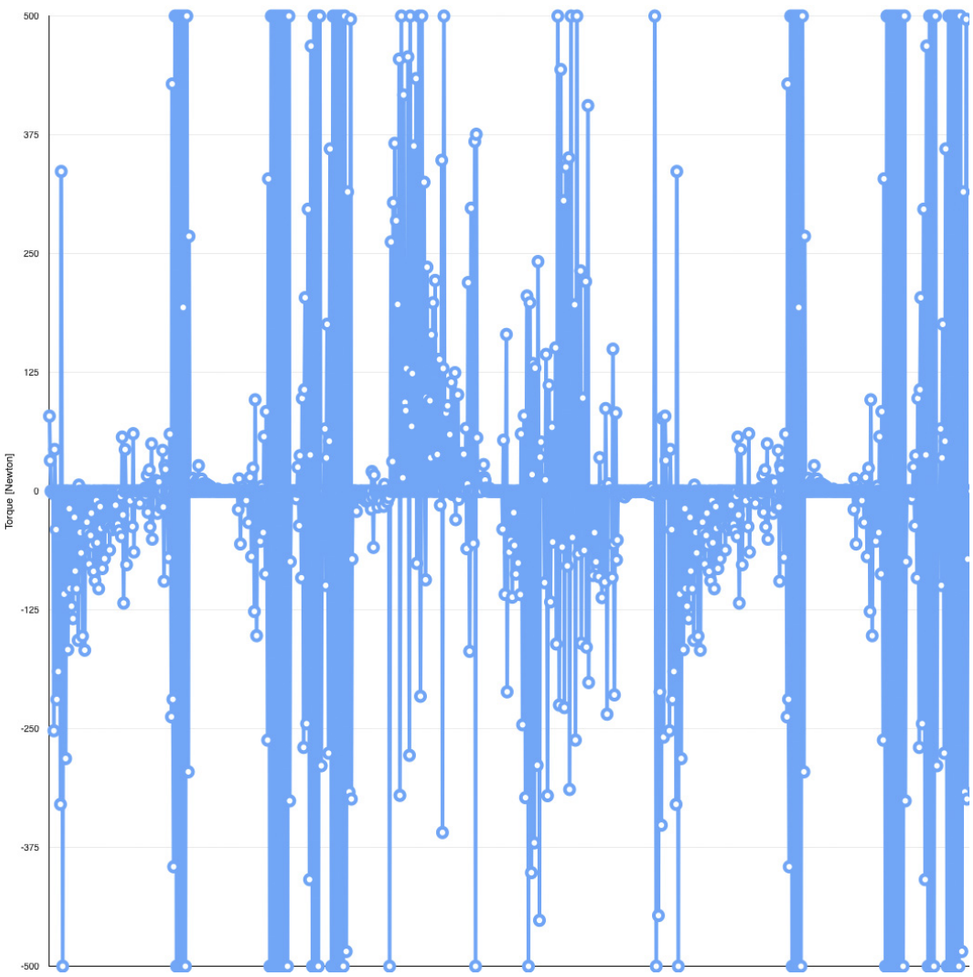


Figure 7.152: Uniform Movement Base Torque, Ki = 350

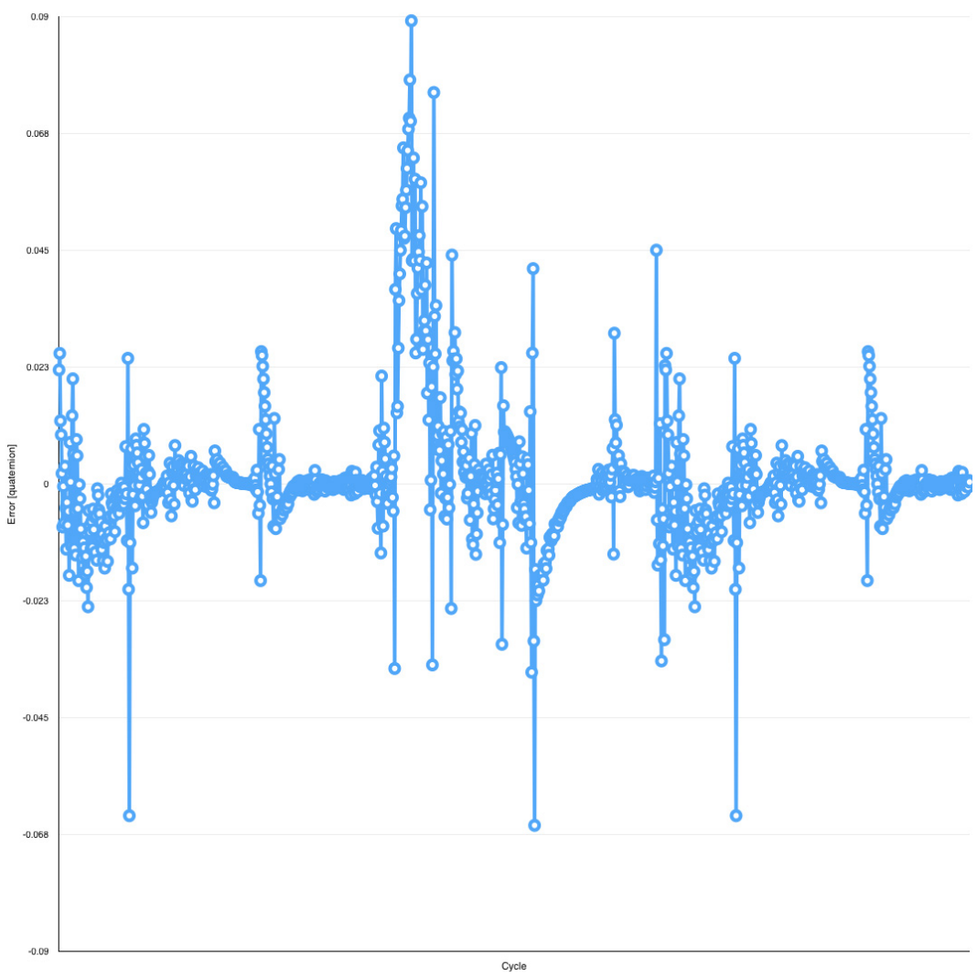


Figure 7.153: Uniform Movement Base Position error, Ki = 400

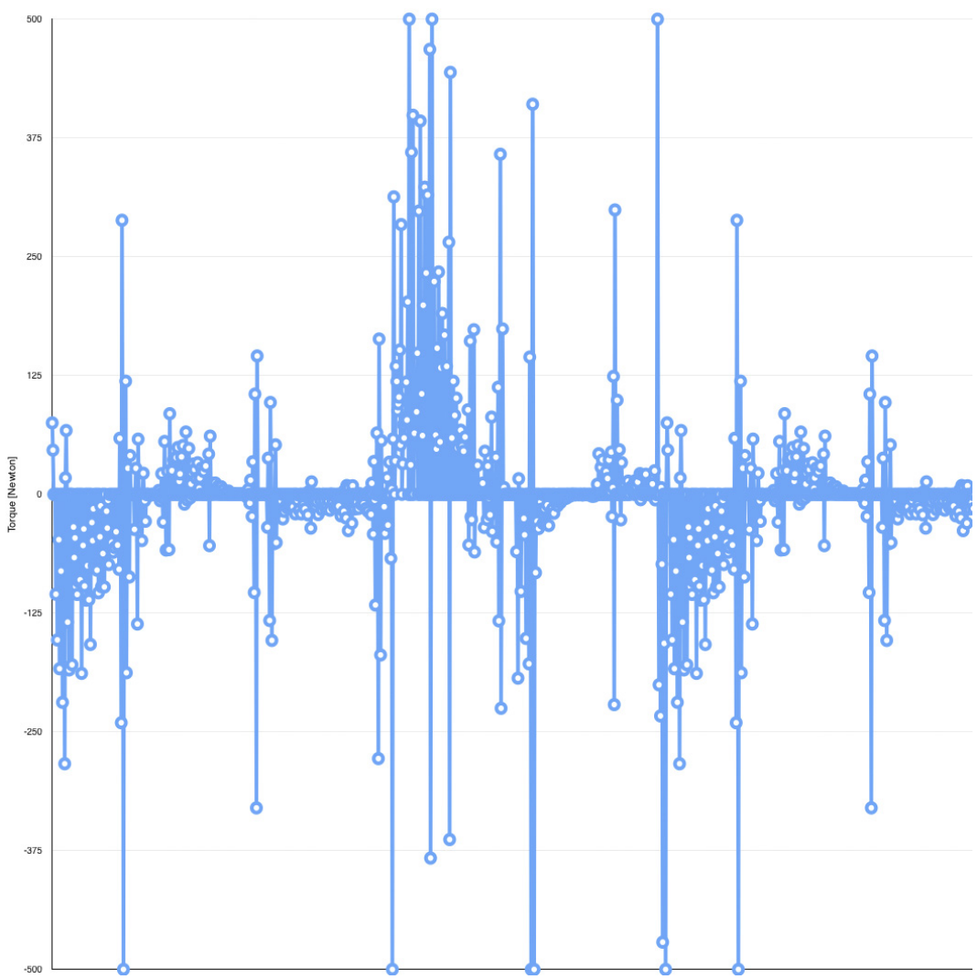


Figure 7.154: Uniform Movement Base Torque, Ki = 400

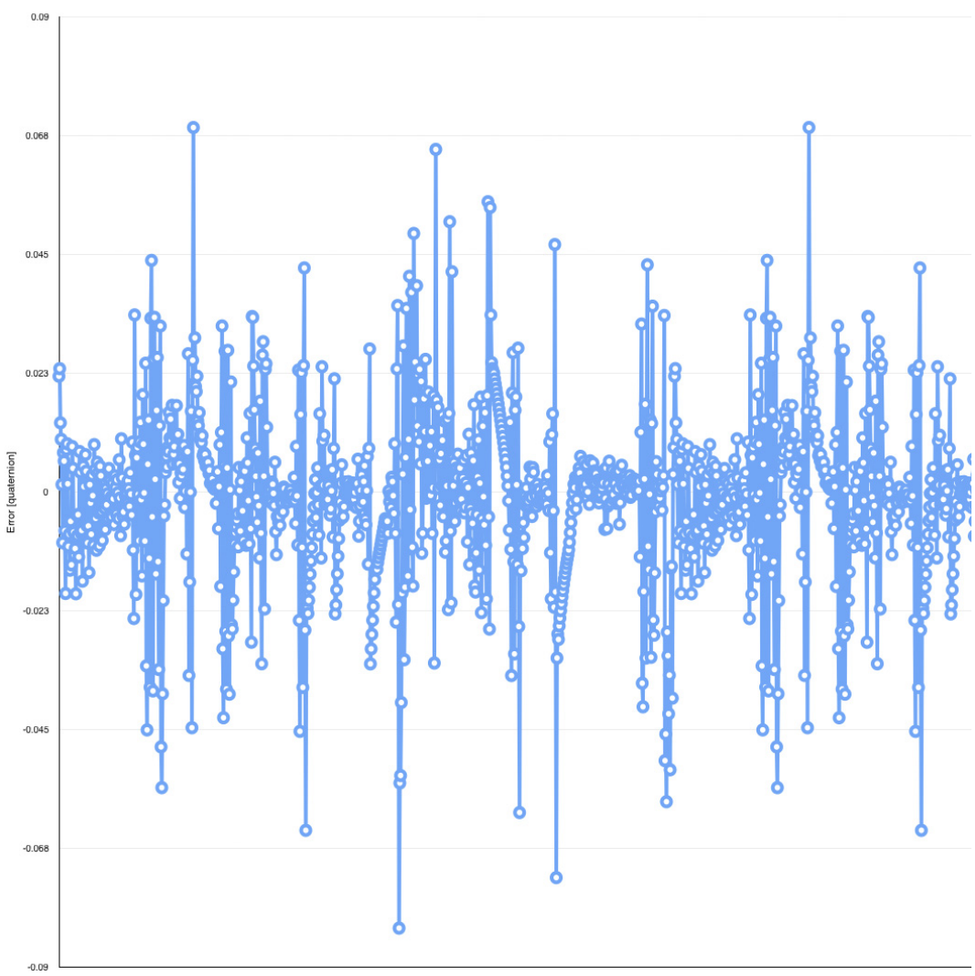


Figure 7.155: Uniform Movement Base Position error, Ki = 450

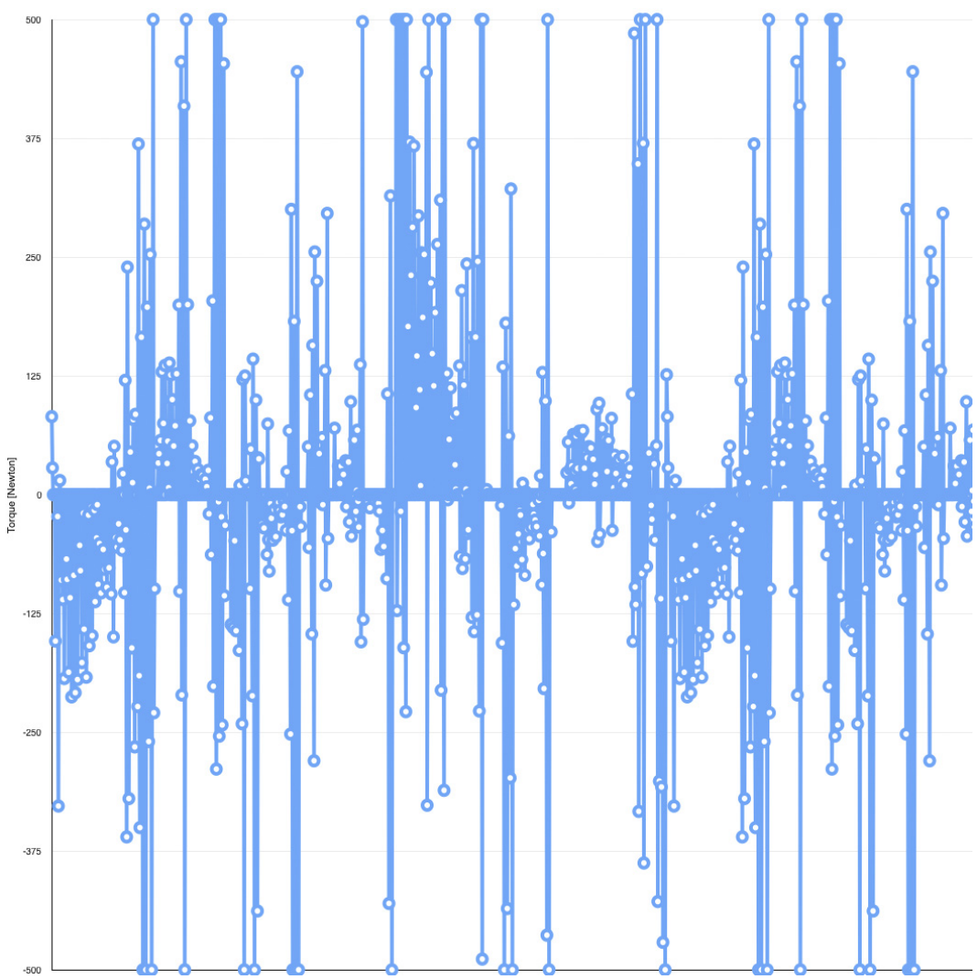


Figure 7.156: Uniform Movement Base Torque, Ki = 450

7.1.3 Derivative variable

7.1.3.1 Joint 1

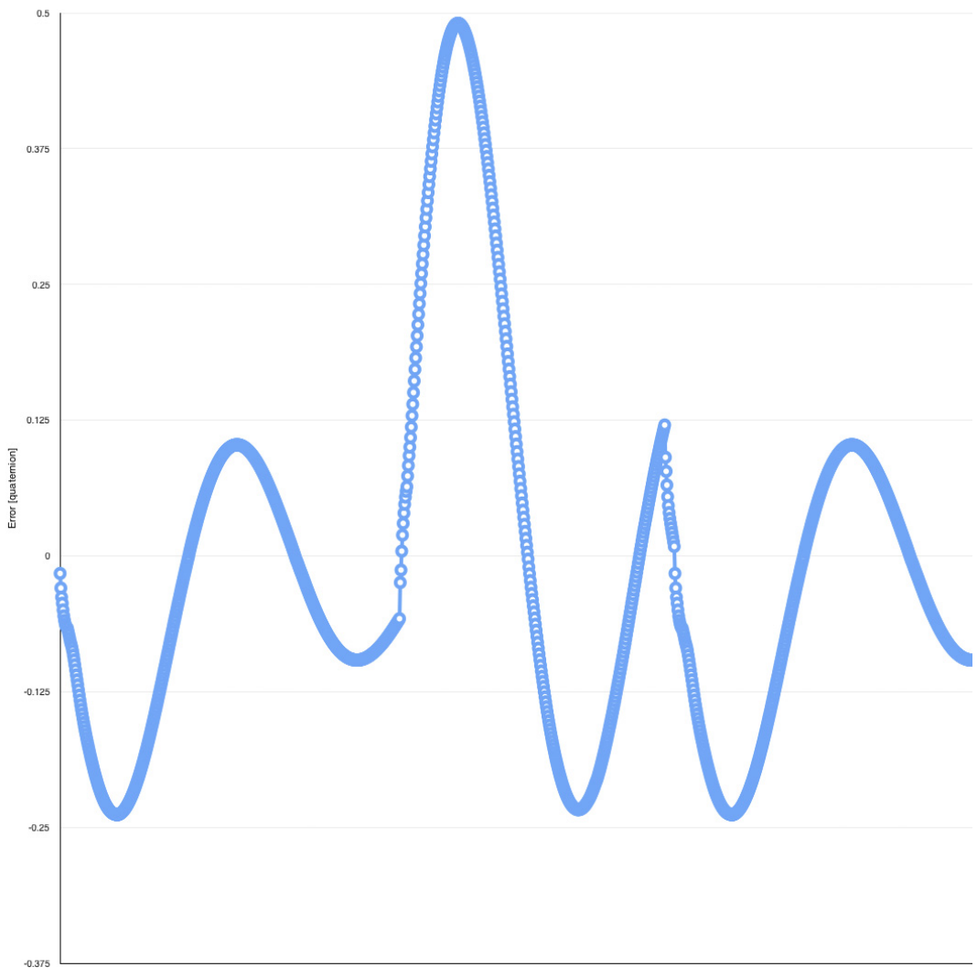


Figure 7.157: Uniform Movement Base Position error, Kd = 100

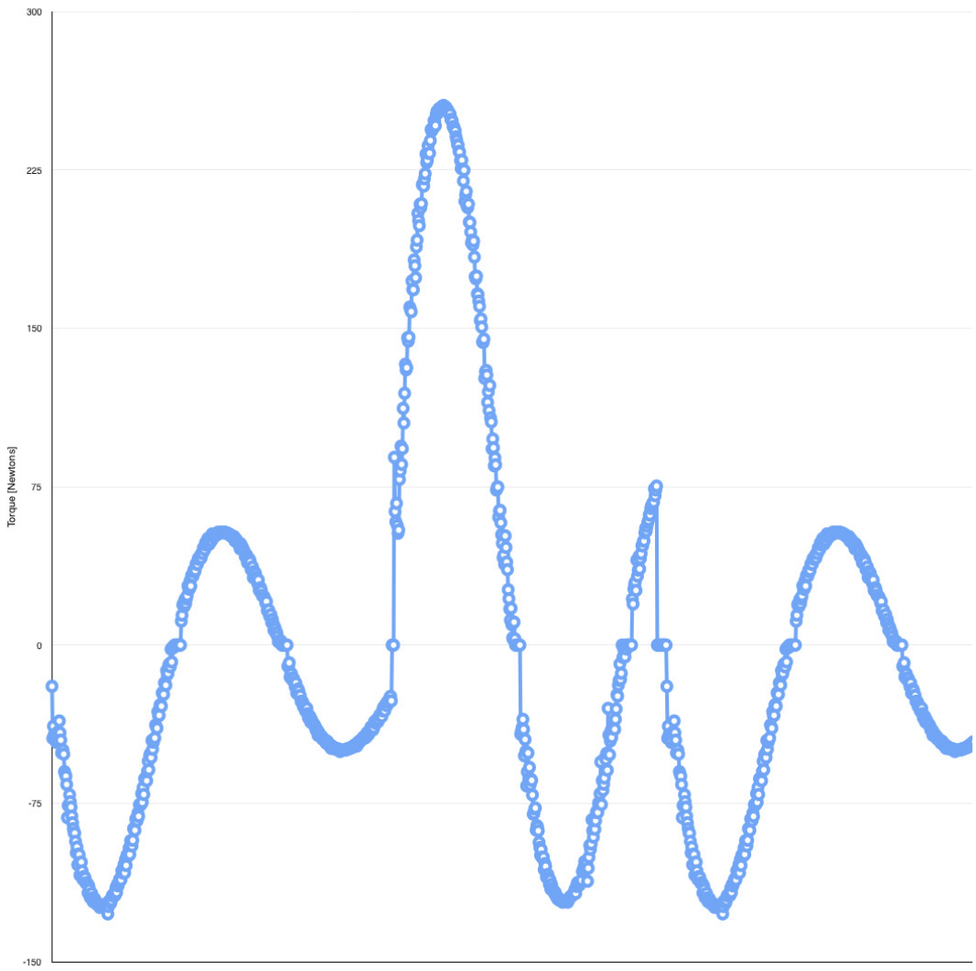


Figure 7.158: Uniform Movement Base Torque, Kd = 100

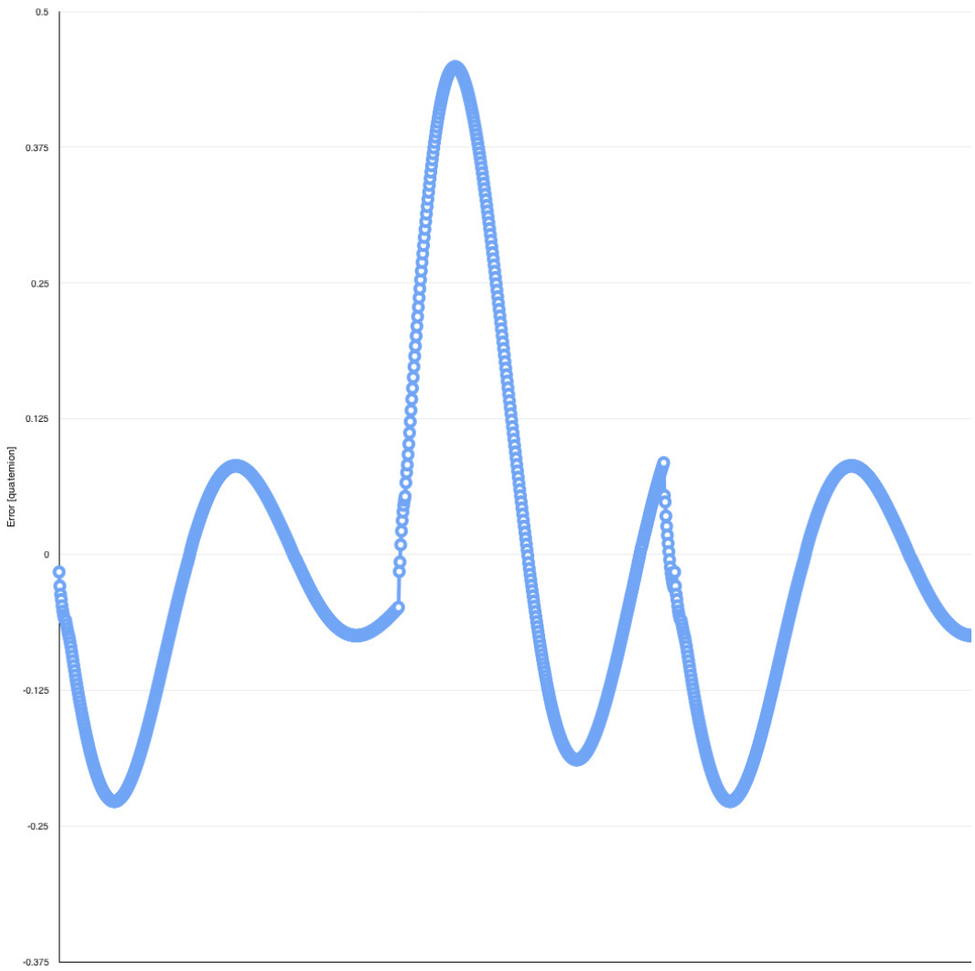


Figure 7.159: Uniform Movement Base Position error, Kd = 150

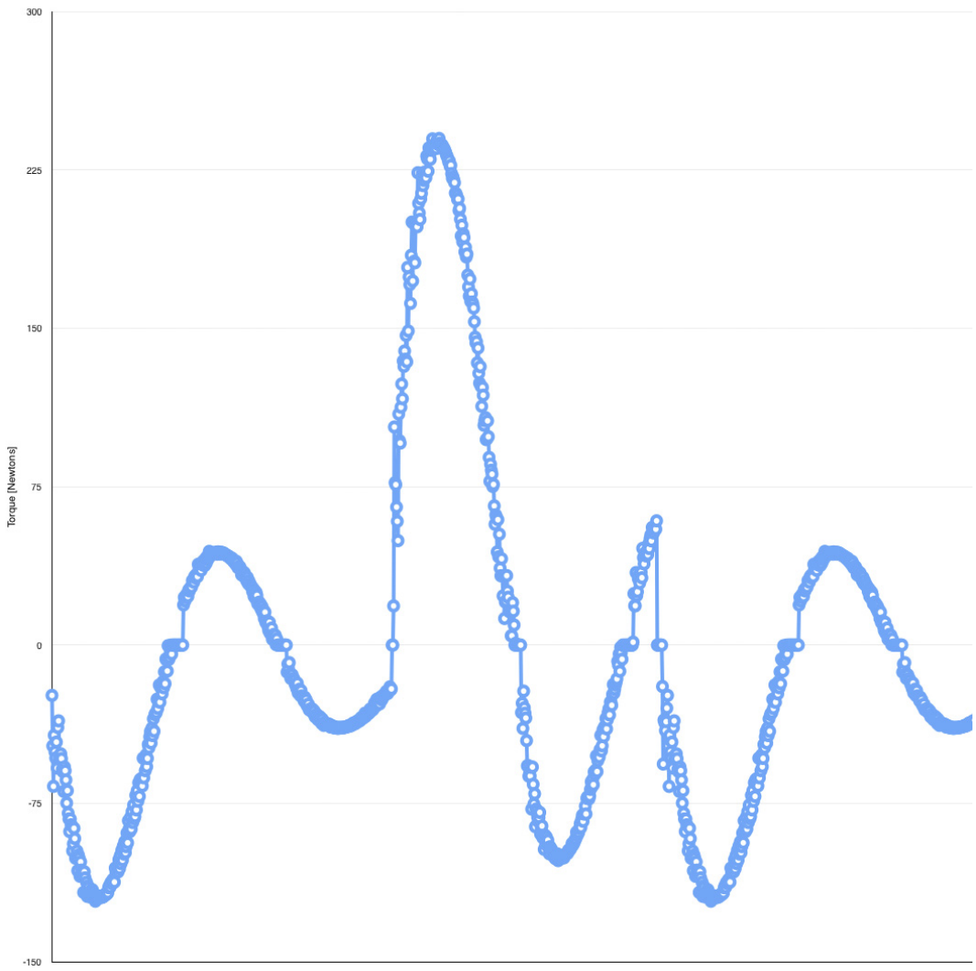


Figure 7.160: Uniform Movement Base Torque, Kd = 150

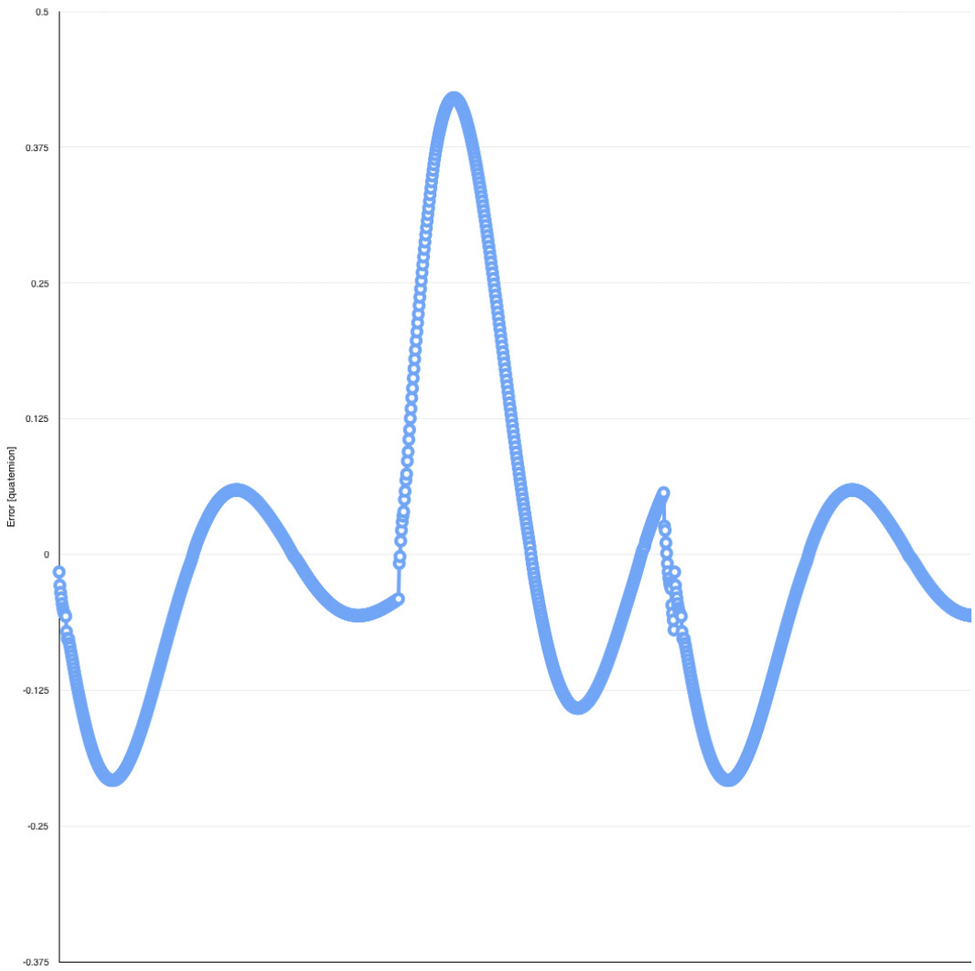


Figure 7.161: Uniform Movement Base Position error, Kd = 200

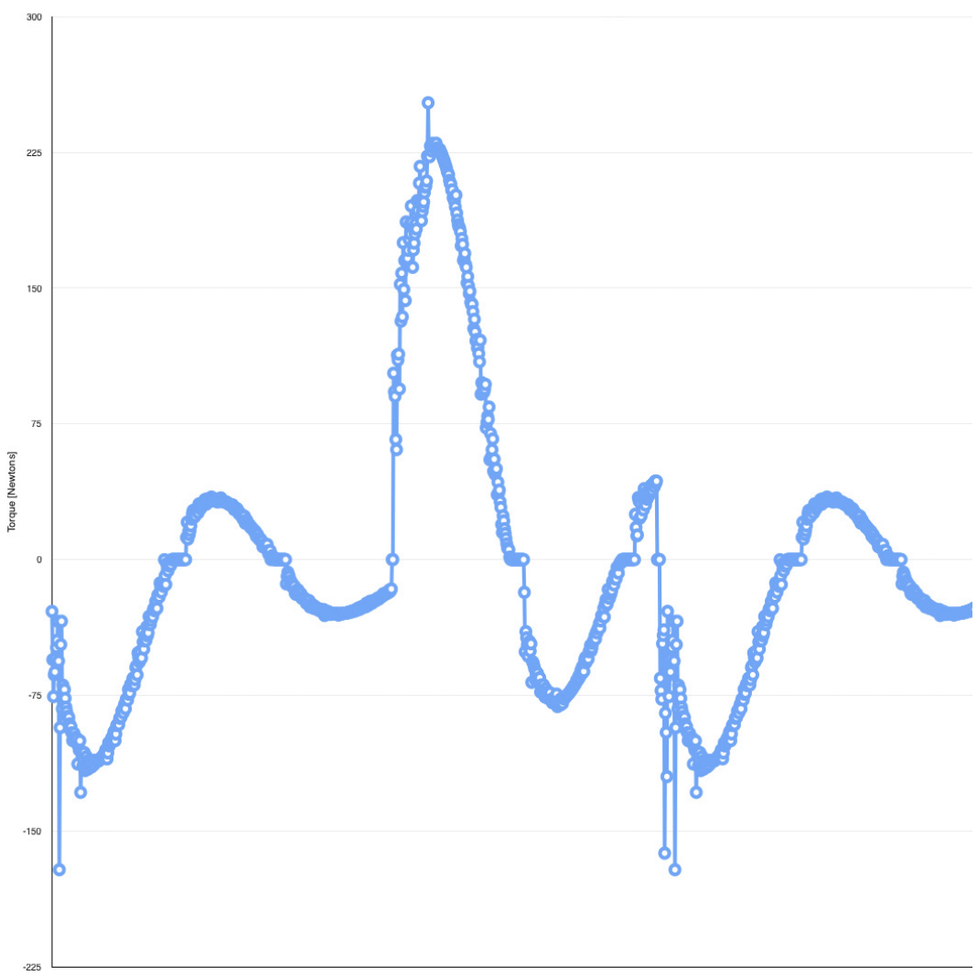


Figure 7.162: Uniform Movement Base Torque, Kd = 200

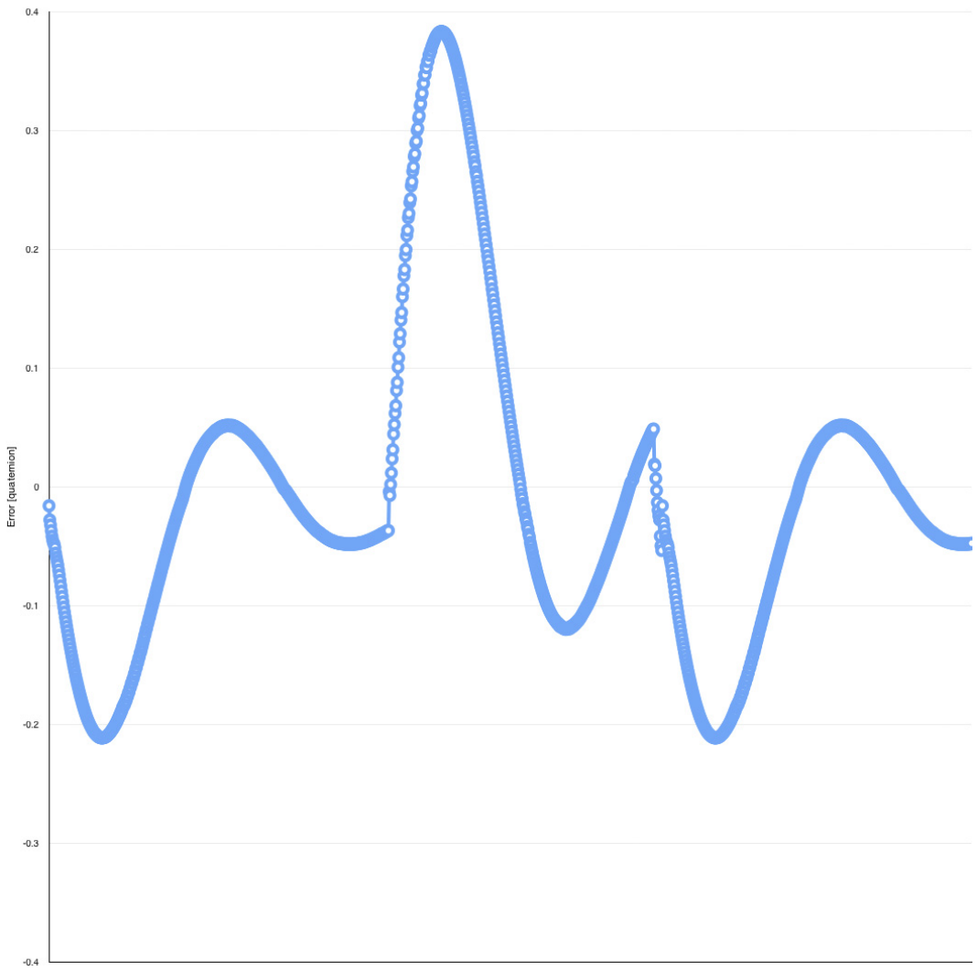


Figure 7.163: Uniform Movement Base Position error, Kd = 250

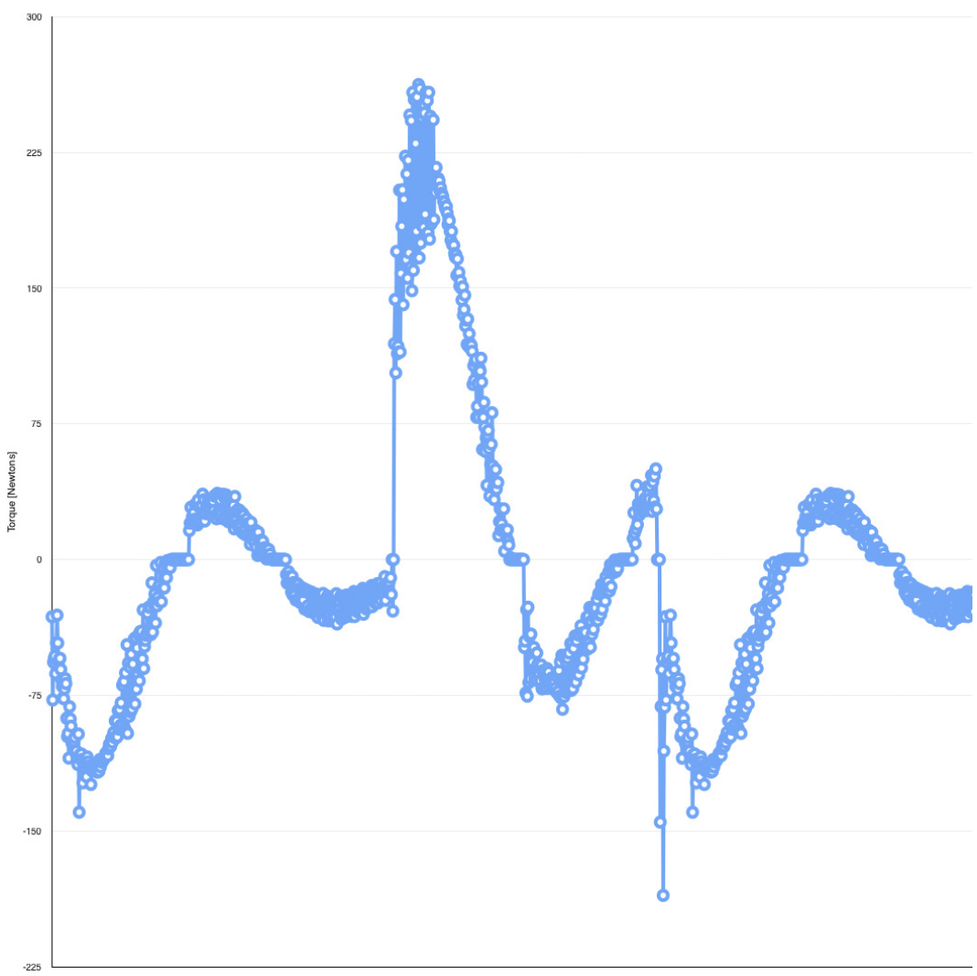


Figure 7.164: Uniform Movement Base Torque, Kd = 250

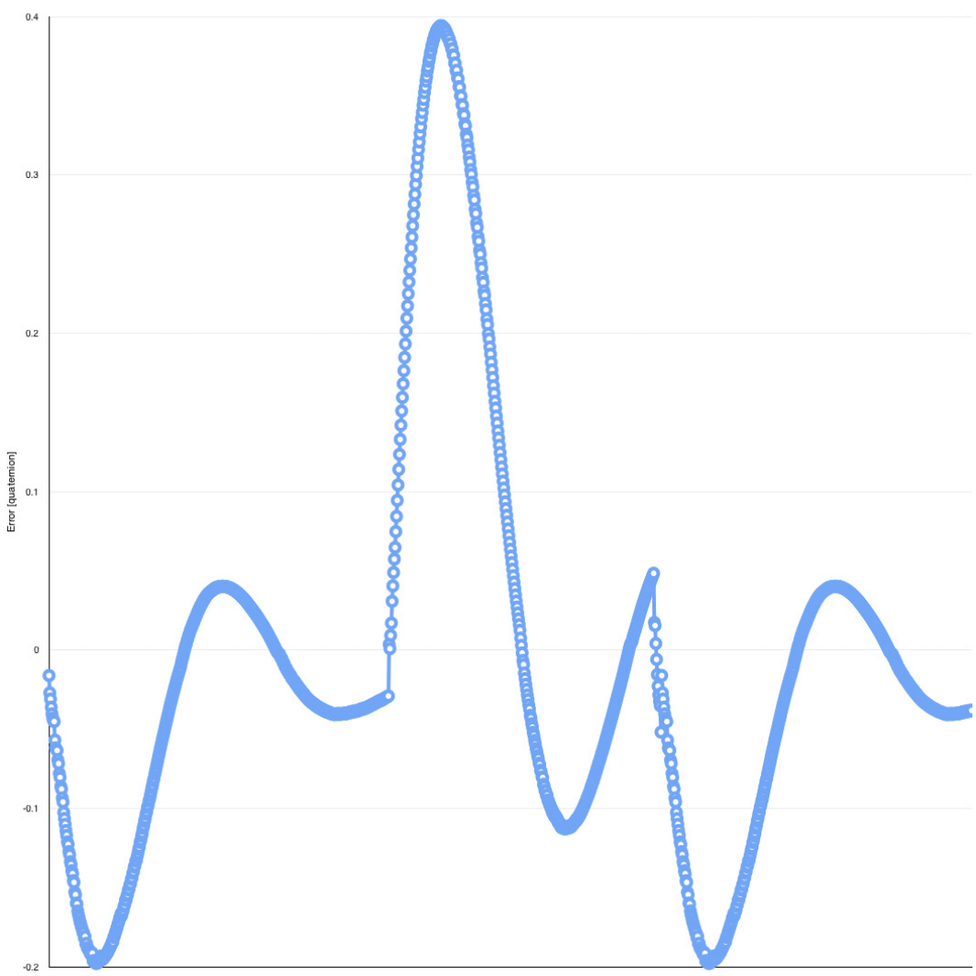


Figure 7.165: Uniform Movement Base Position error, Kd = 300

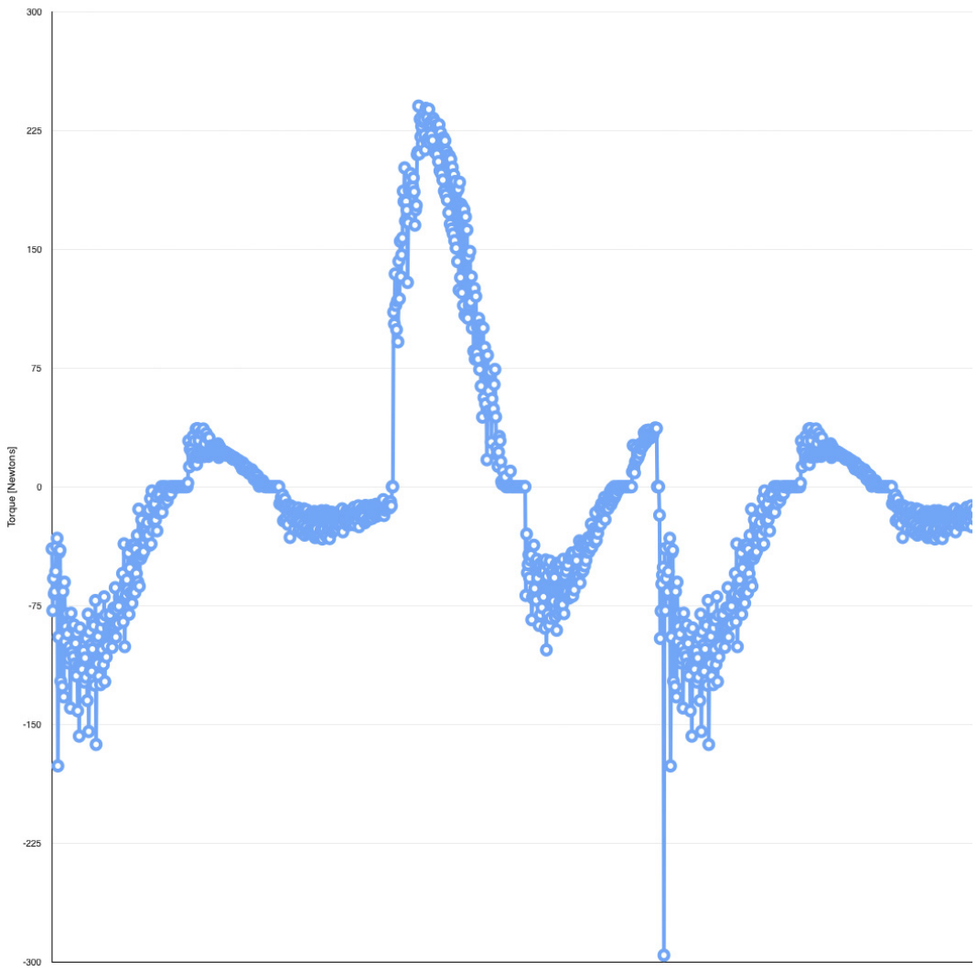


Figure 7.166: Uniform Movement Base Torque, Kd = 300

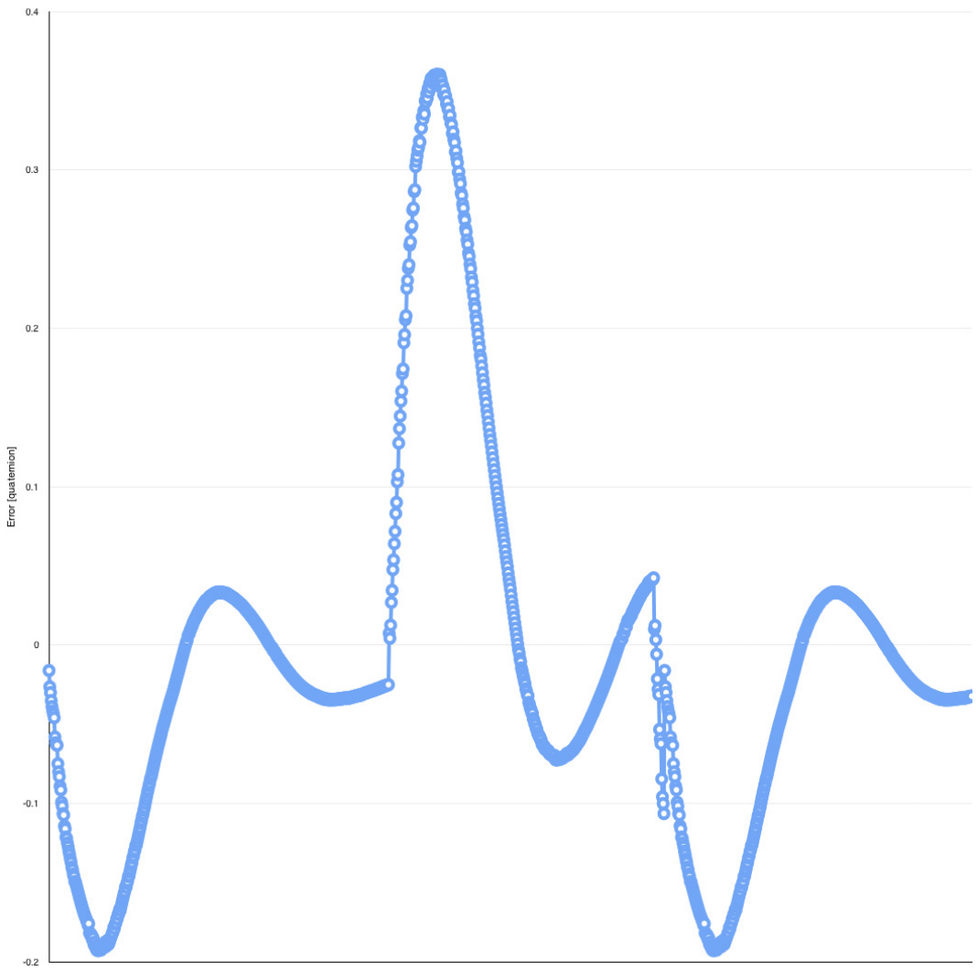


Figure 7.167: Uniform Movement Base Position error, Kd = 350

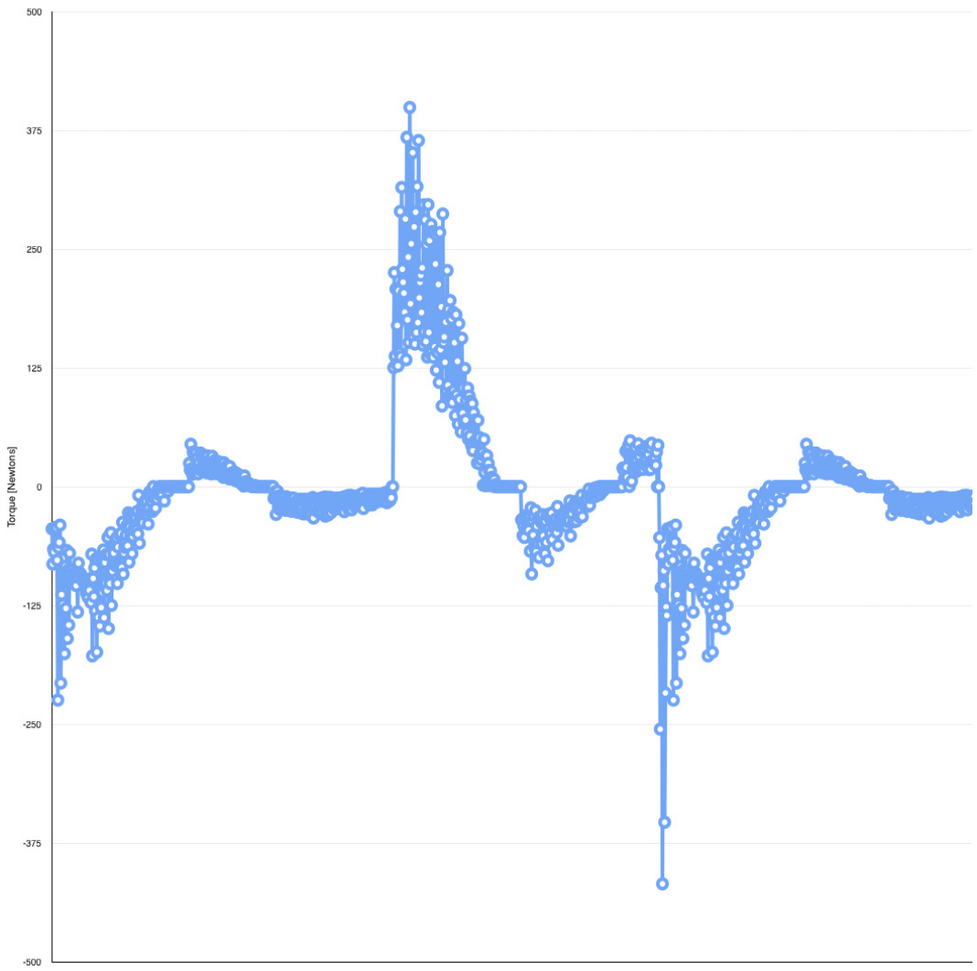


Figure 7.168: Uniform Movement Base Torque, Kd = 350

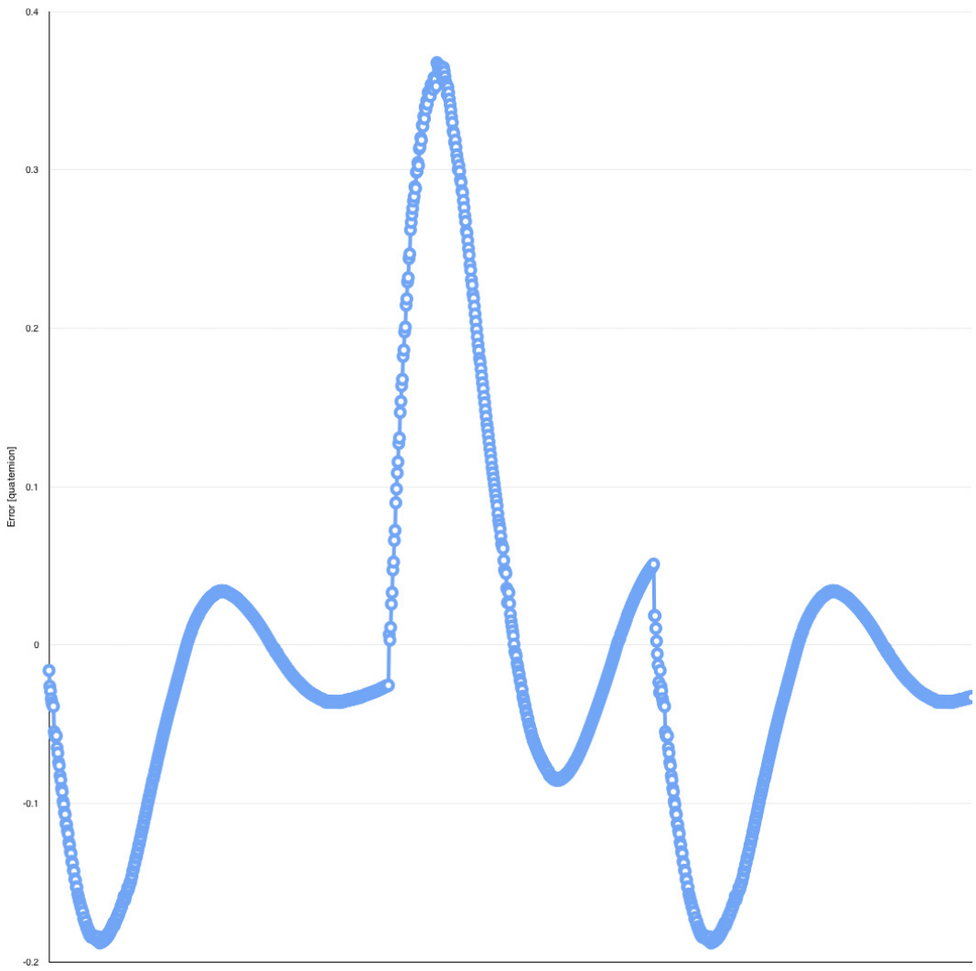


Figure 7.169: Uniform Movement Base Position error, Kd = 400

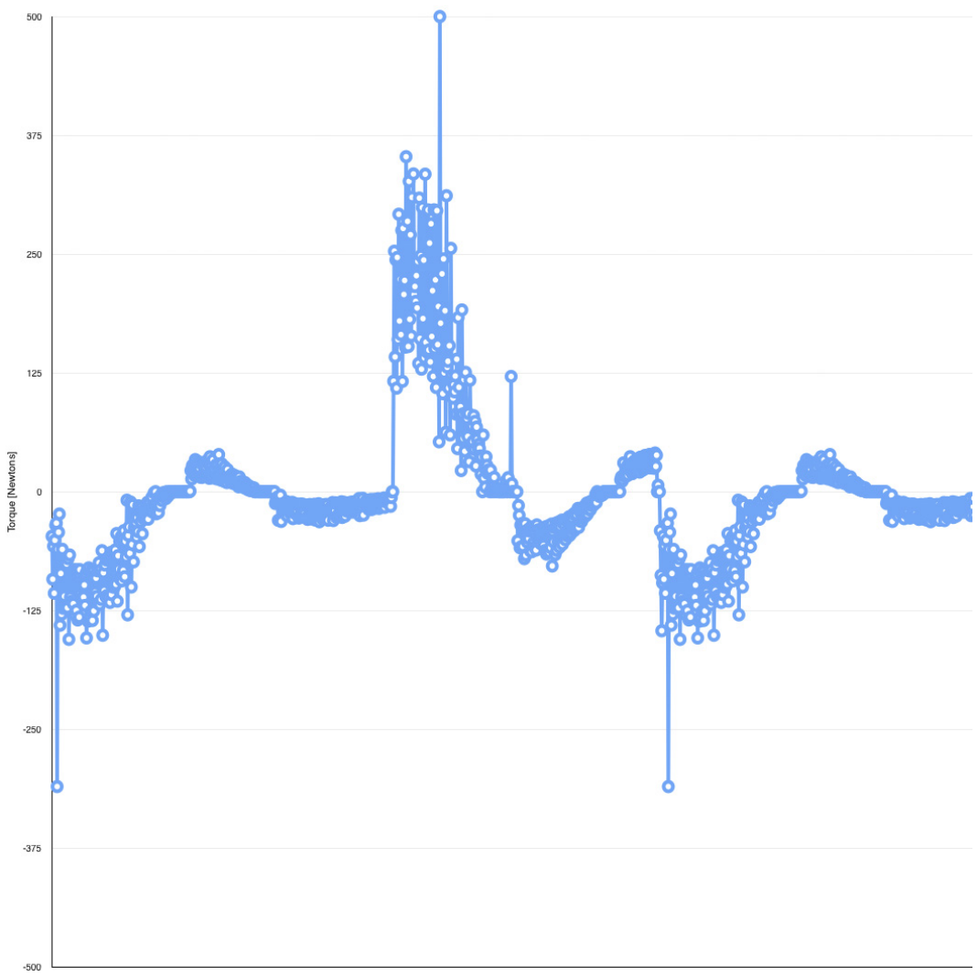


Figure 7.170: Uniform Movement Base Torque, Kd = 400

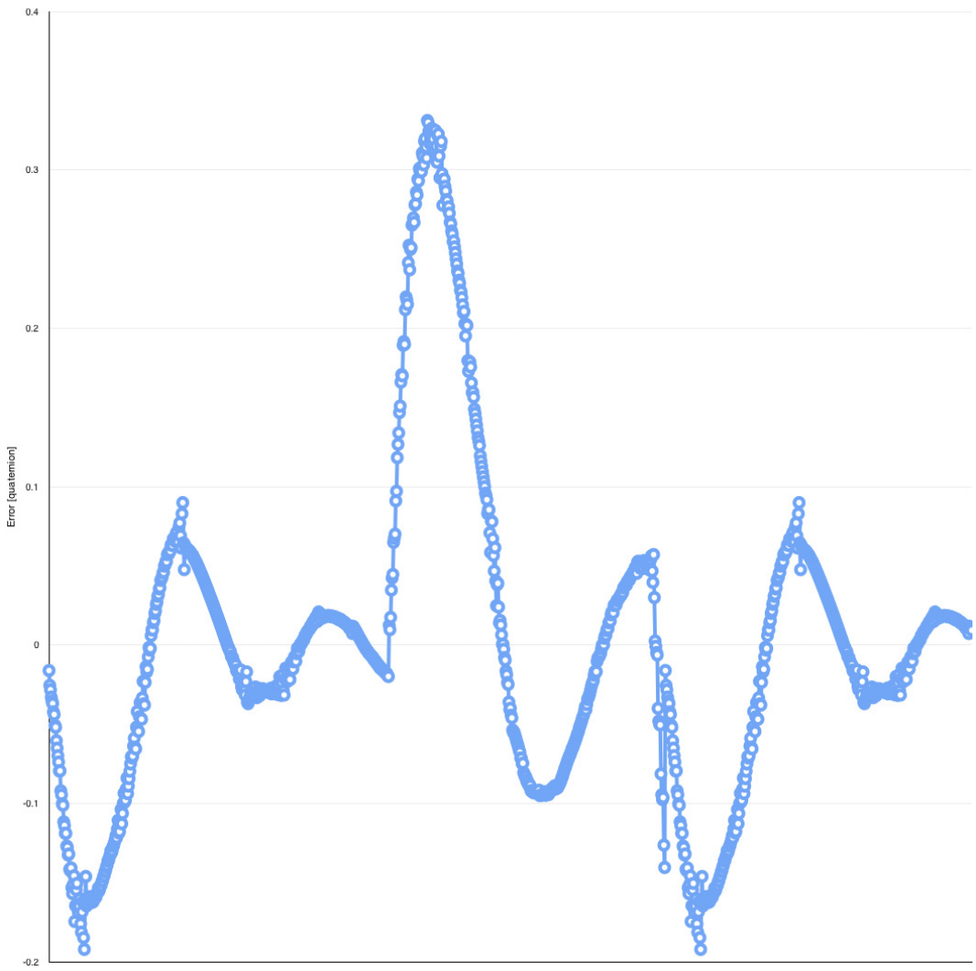


Figure 7.171: Uniform Movement Base Position error, Kd = 450

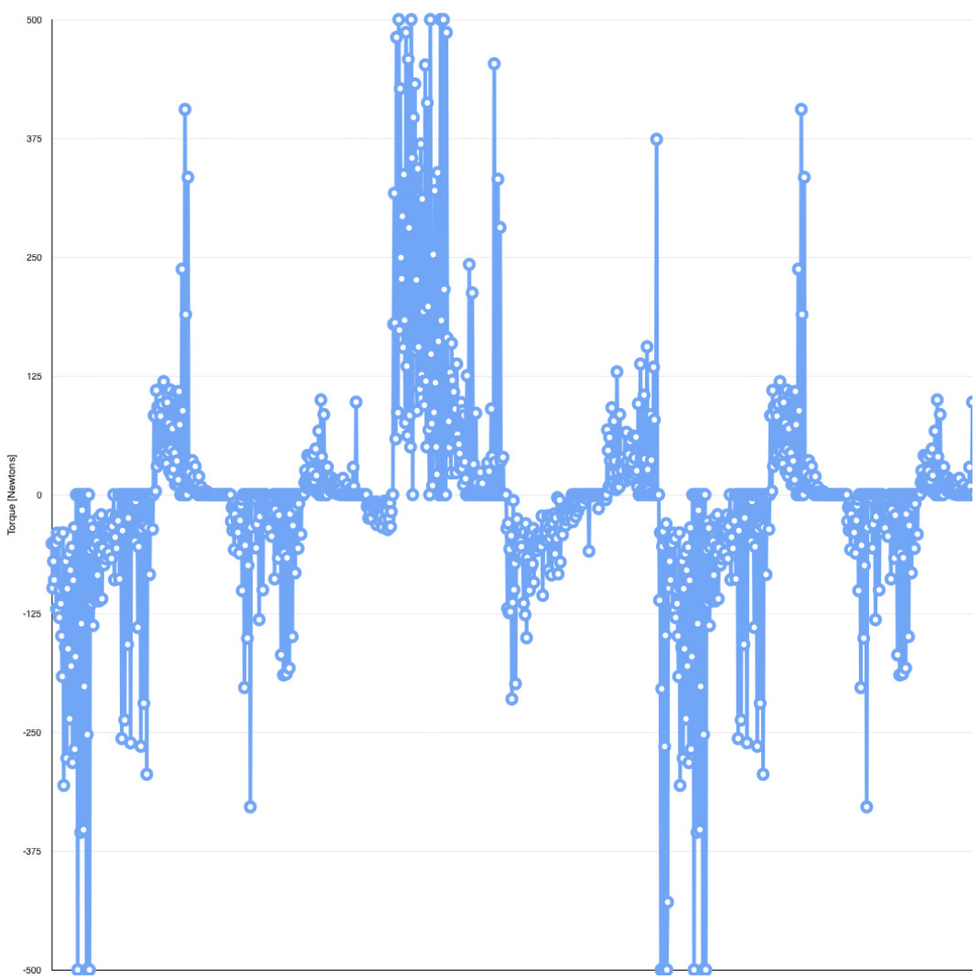


Figure 7.172: Uniform Movement Base Torque, Kd = 450

7.1.3.2 Joint 2

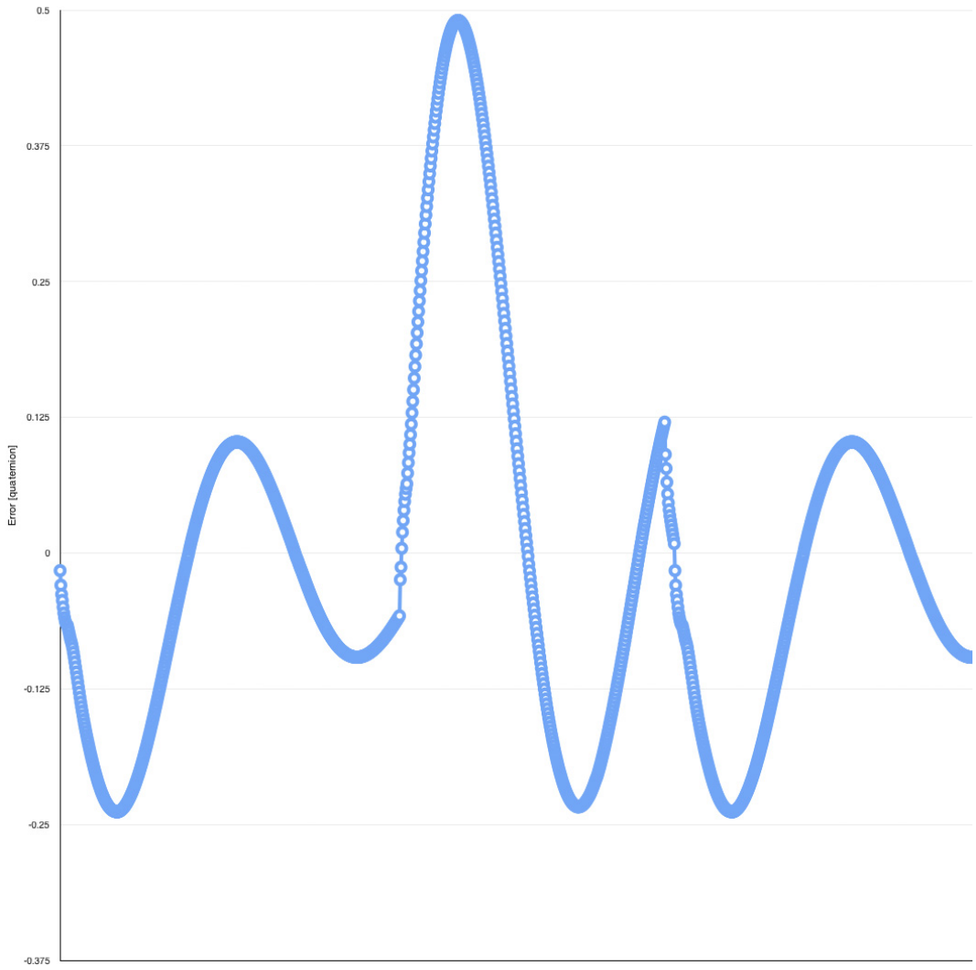


Figure 7.173: Uniform Movement Base Position error, Kd = 100

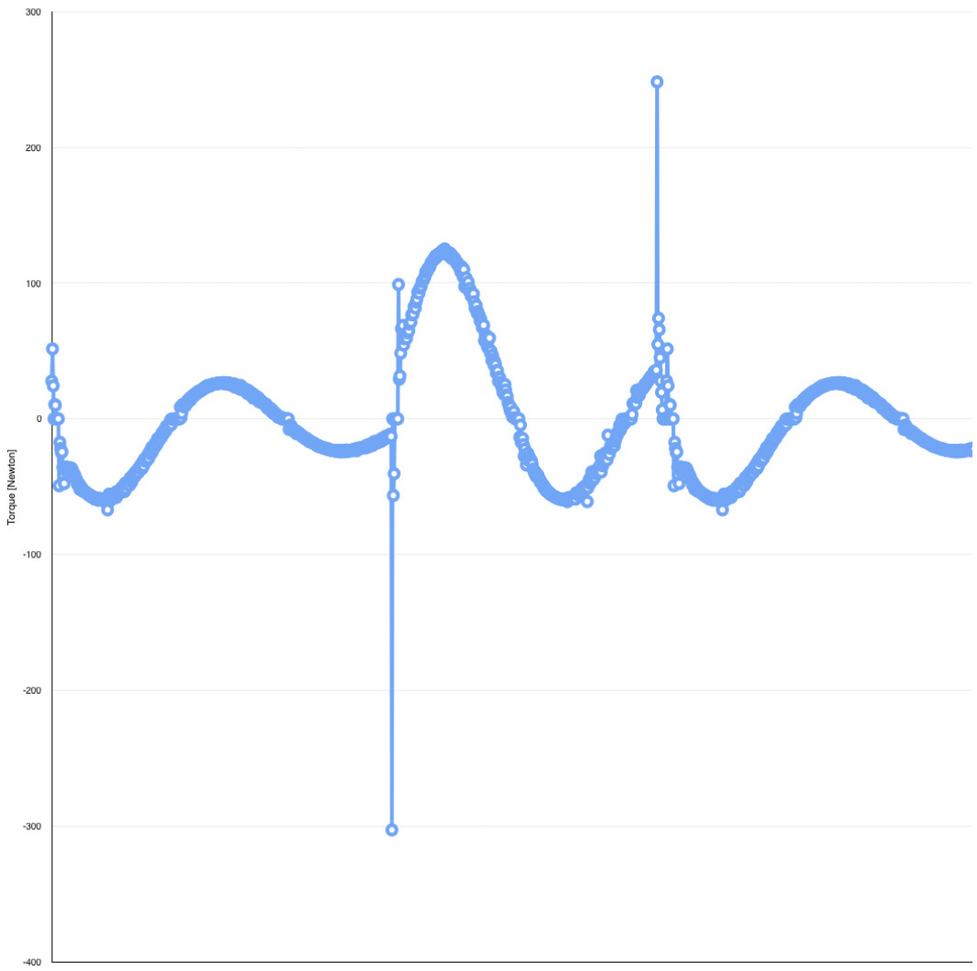


Figure 7.174: Uniform Movement Base Torque, Kd = 100

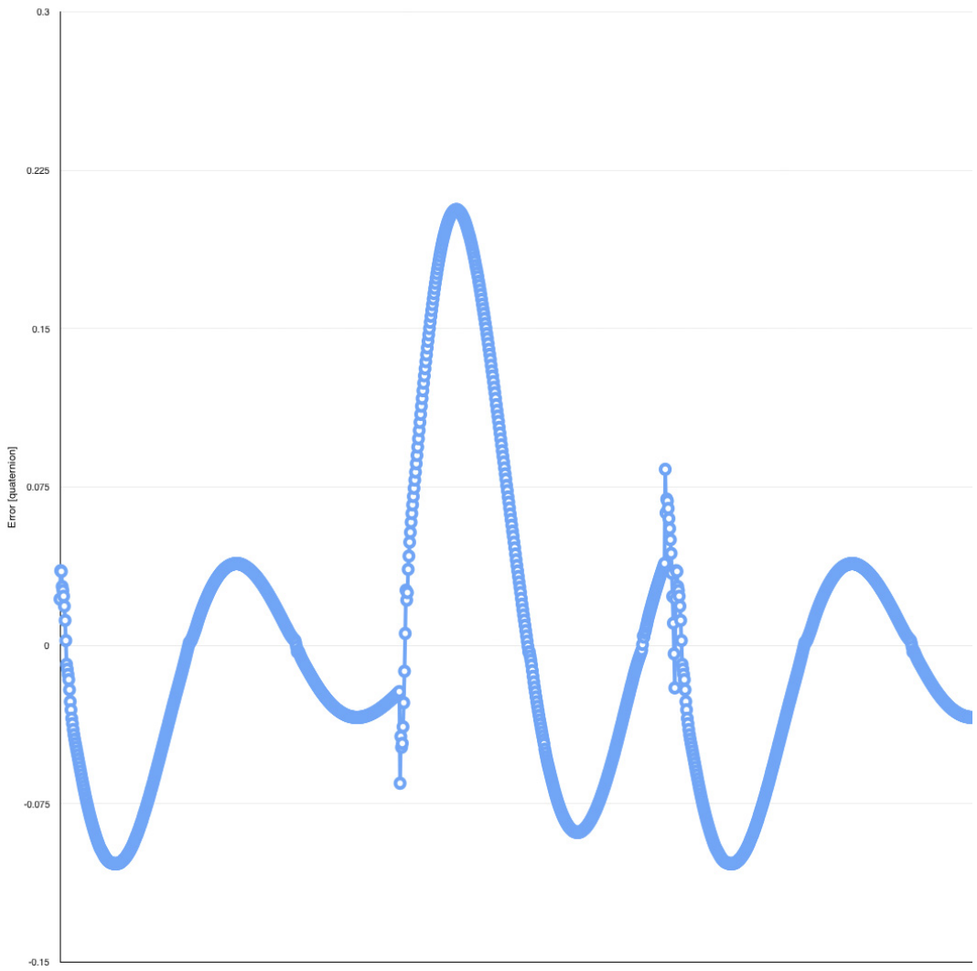


Figure 7.175: Uniform Movement Base Position error, Kd = 150

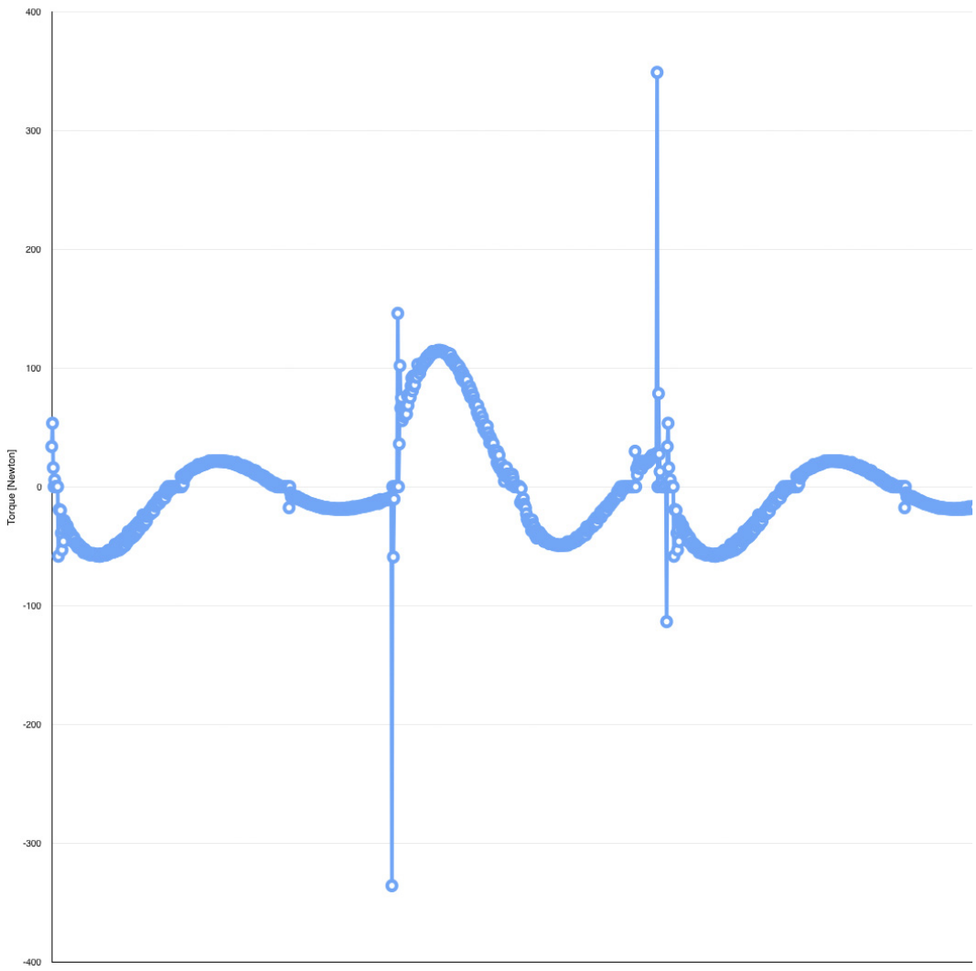


Figure 7.176: Uniform Movement Base Torque, Kd = 150

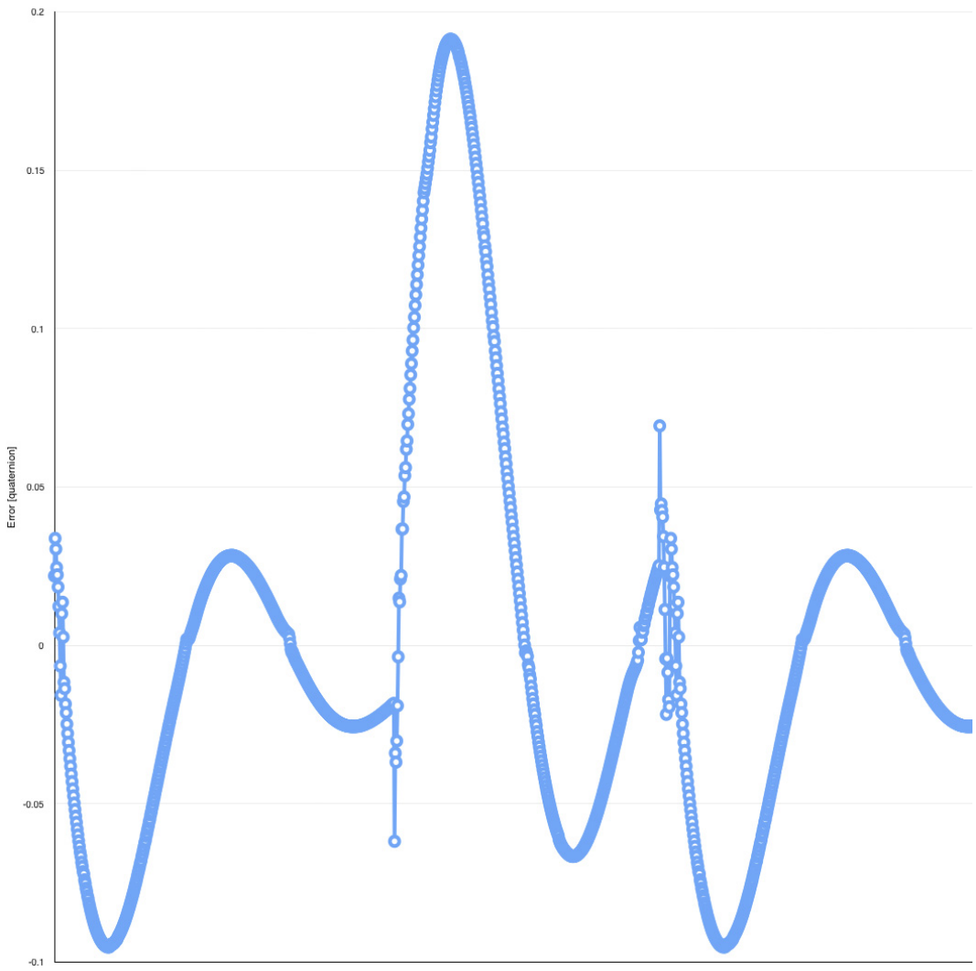


Figure 7.177: Uniform Movement Base Position error, Kd = 200

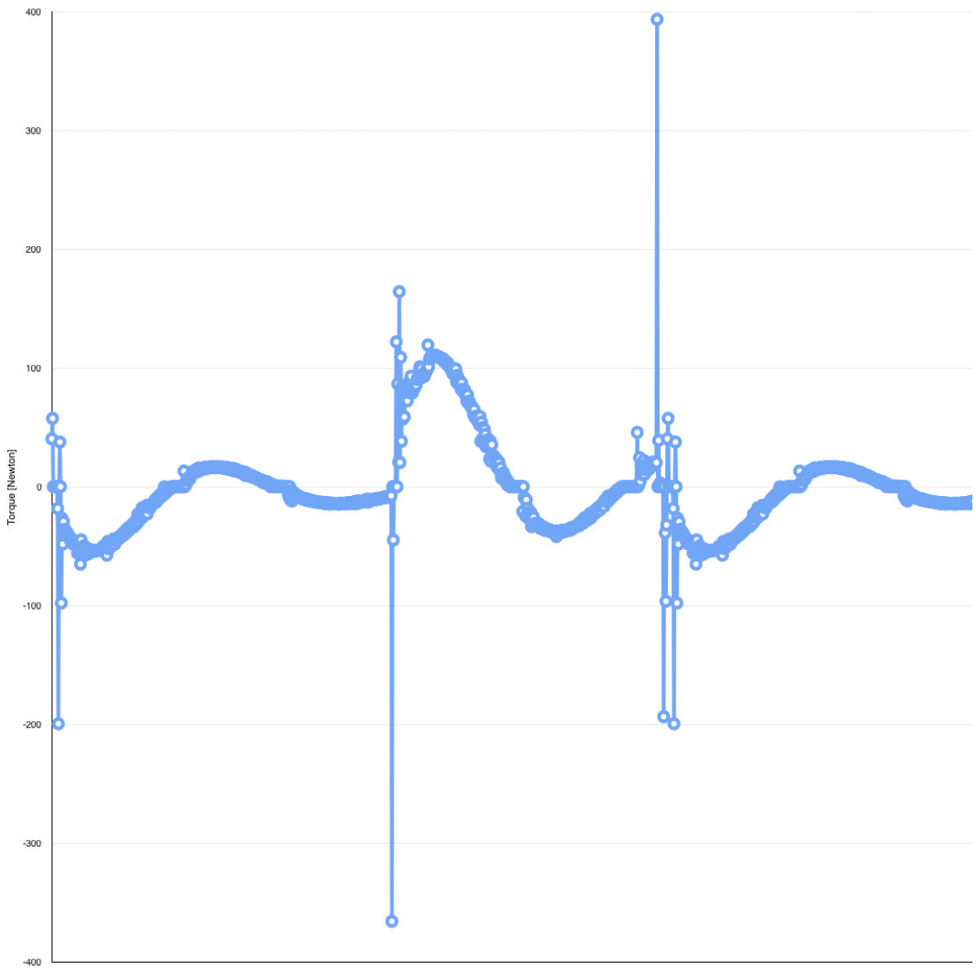


Figure 7.178: Uniform Movement Base Torque, Kd = 200

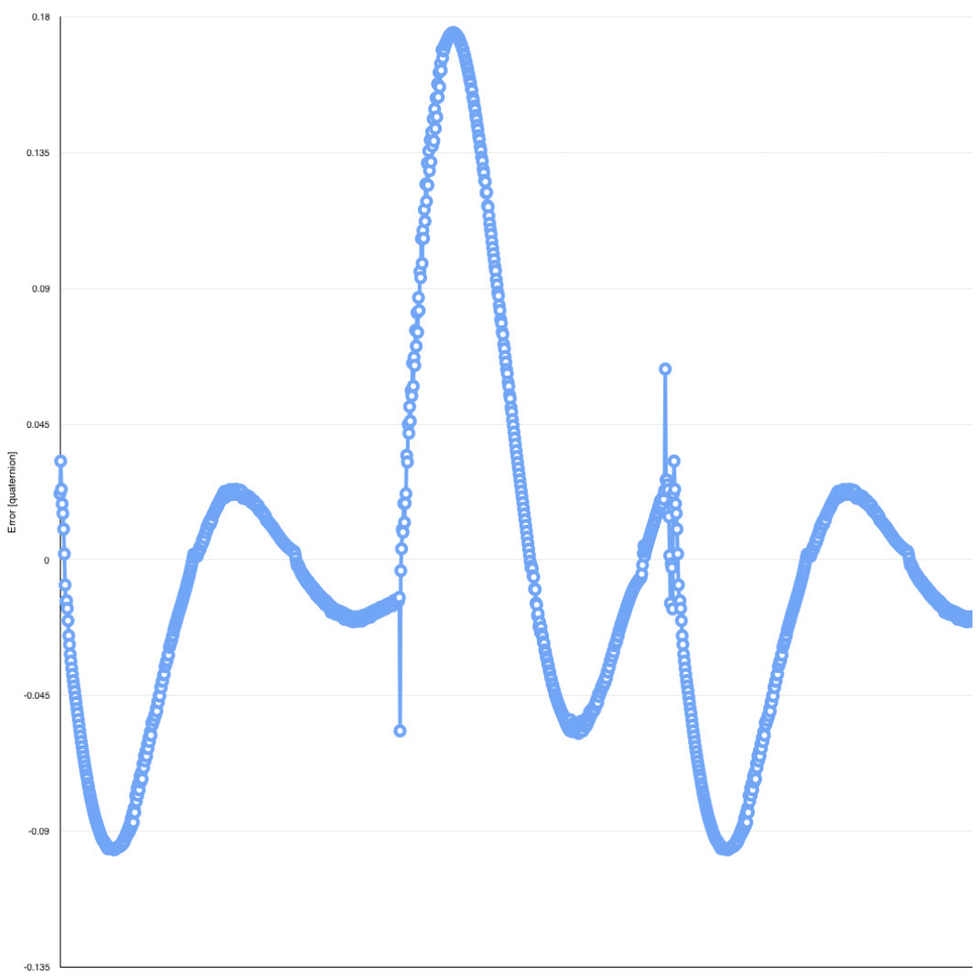


Figure 7.179: Uniform Movement Base Position error, Kd = 250

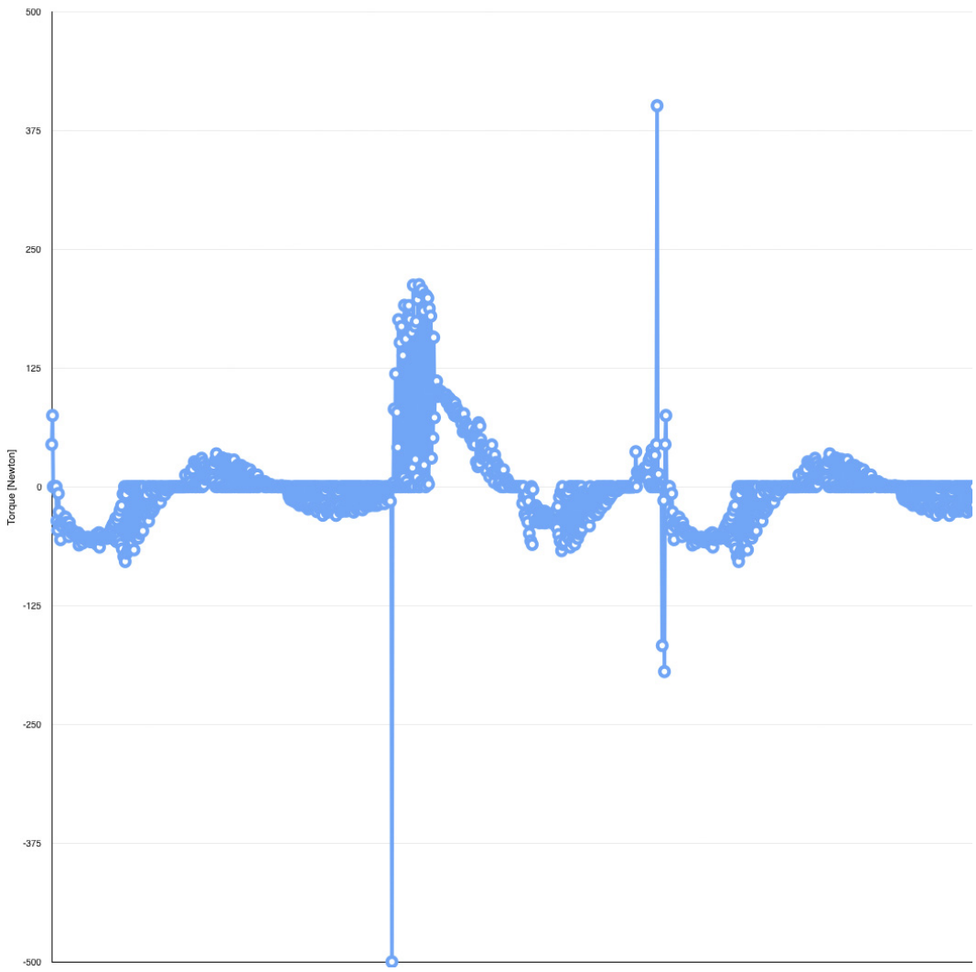


Figure 7.180: Uniform Movement Base Torque, Kd = 250

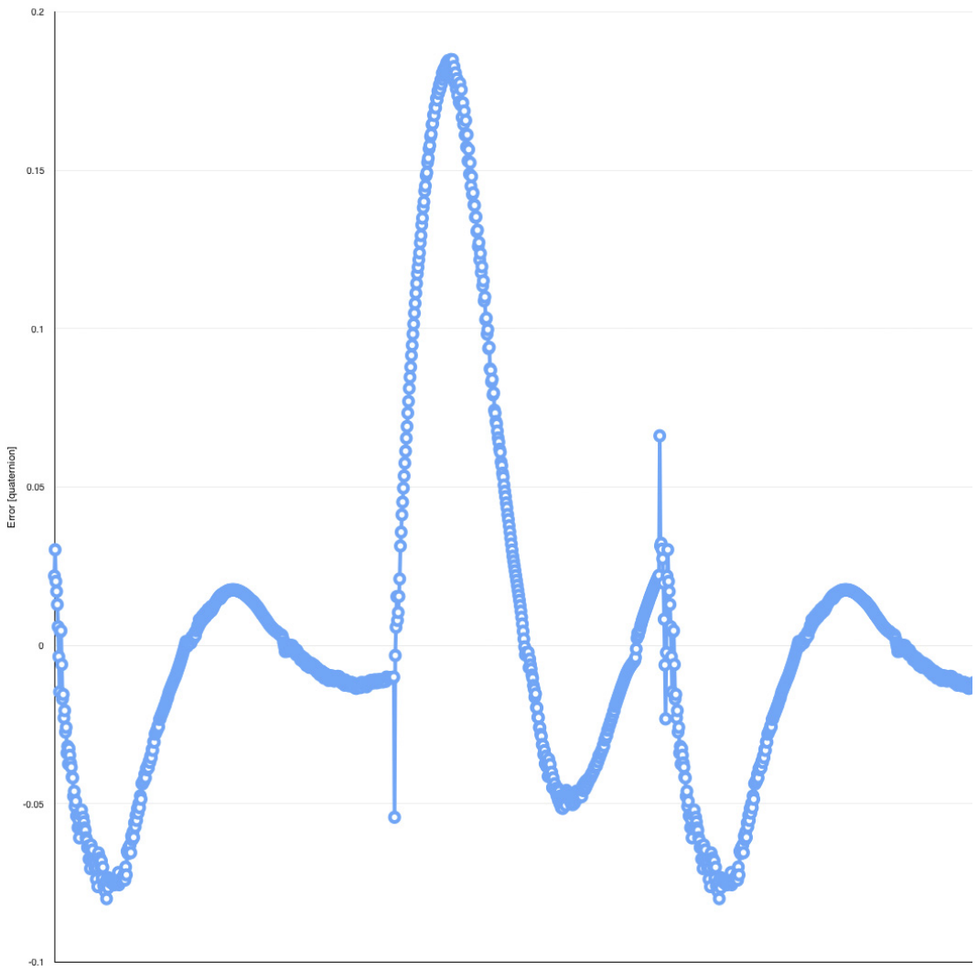


Figure 7.181: Uniform Movement Base Position error, Kd = 300

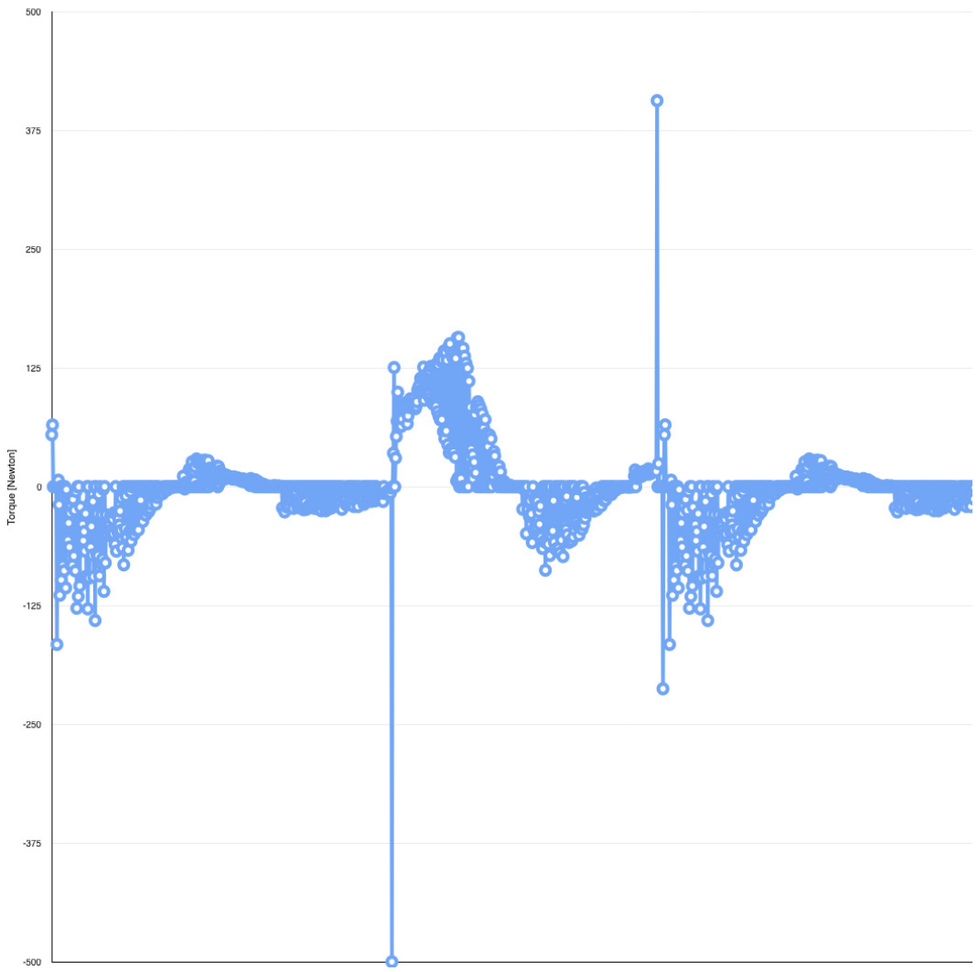


Figure 7.182: Uniform Movement Base Torque, Kd = 300

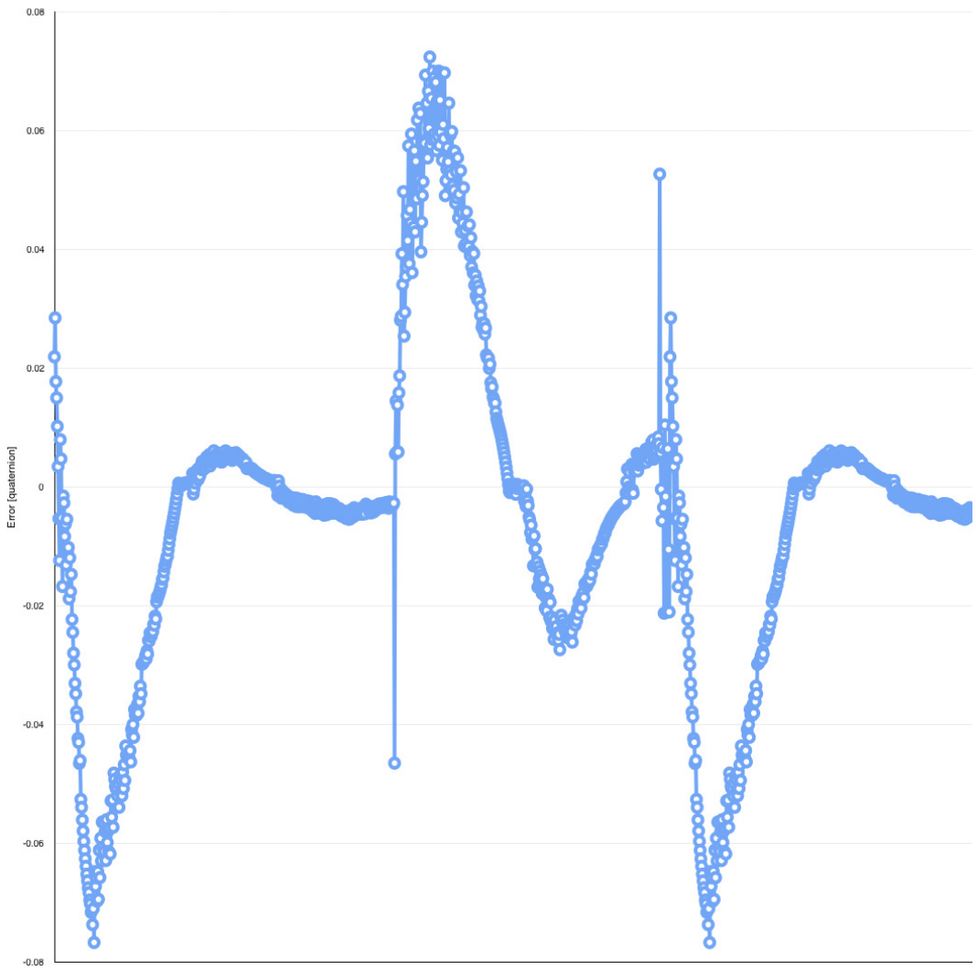


Figure 7.183: Uniform Movement Base Position error, Kd = 350

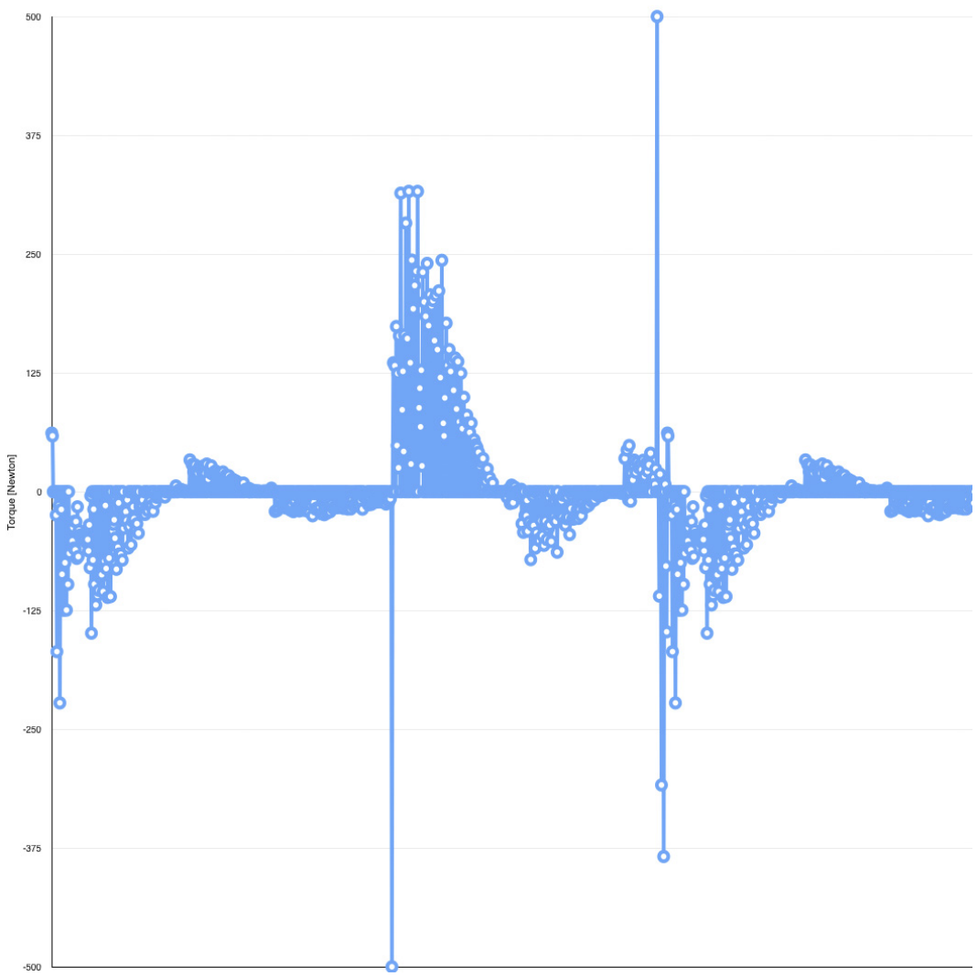


Figure 7.184: Uniform Movement Base Torque, Kd = 350

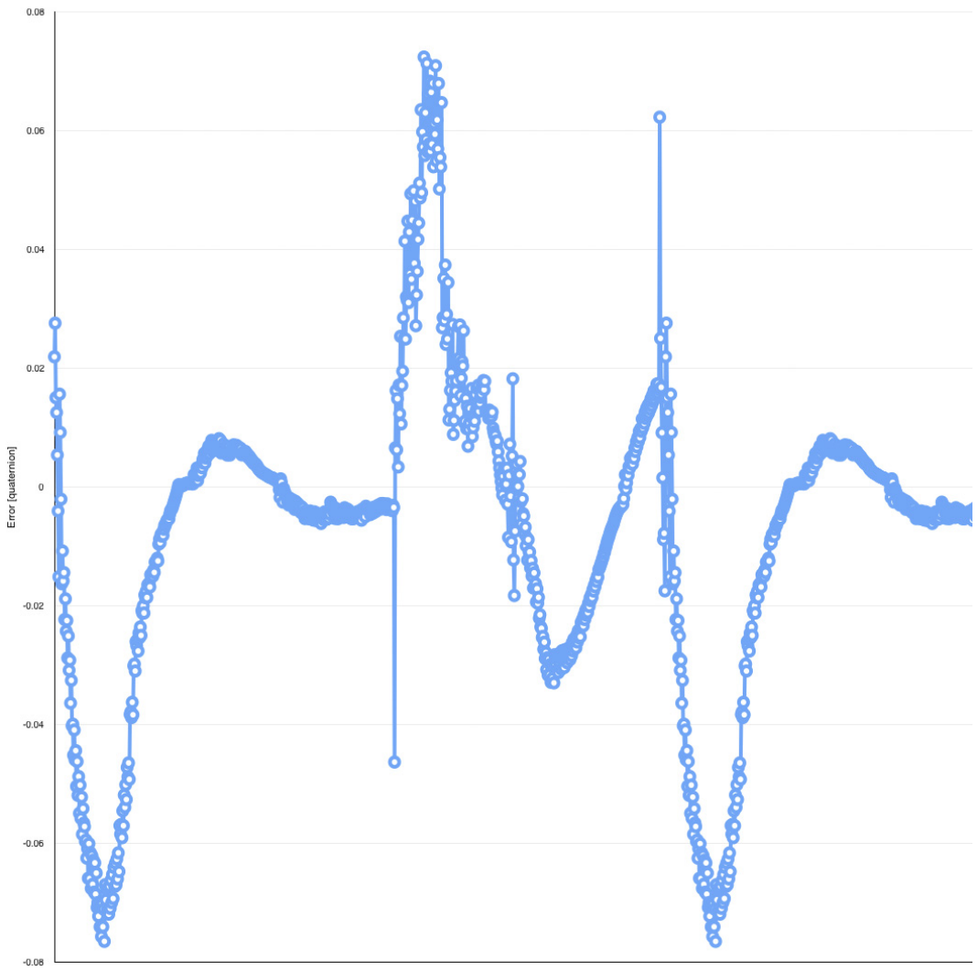


Figure 7.185: Uniform Movement Base Position error, Kd = 400

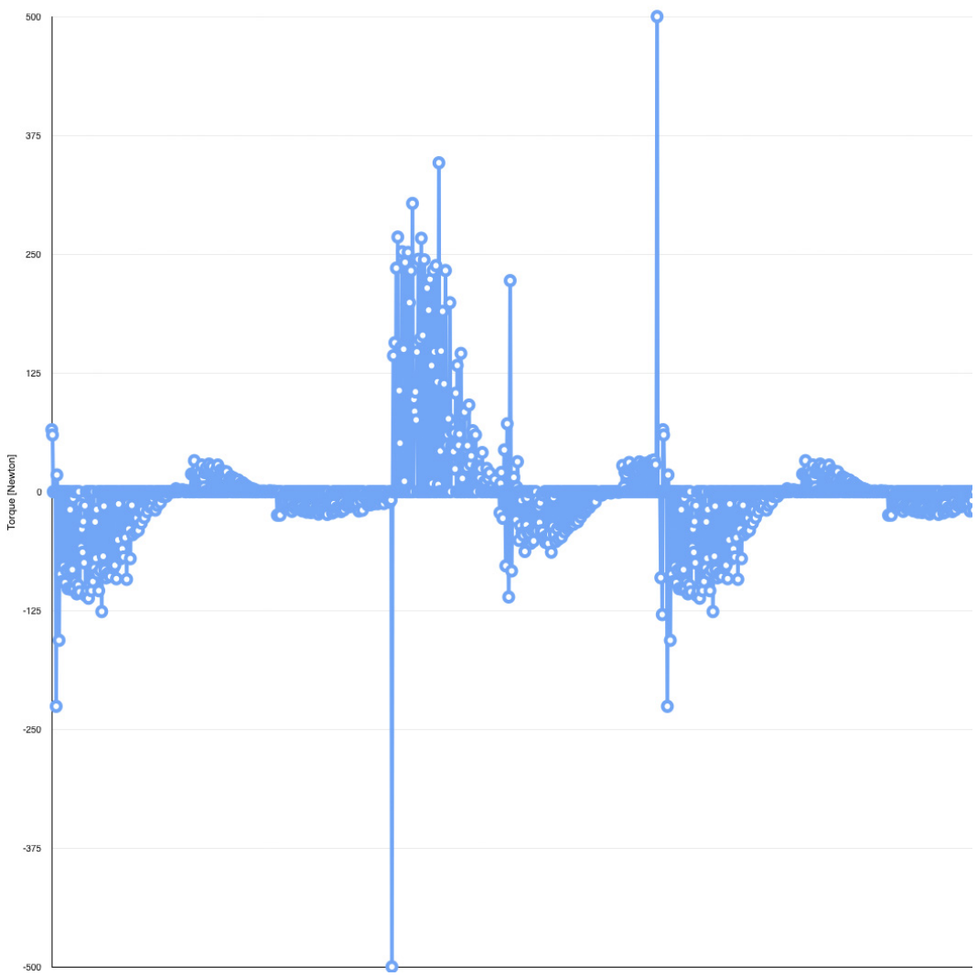


Figure 7.186: Uniform Movement Base Torque, Kd = 400

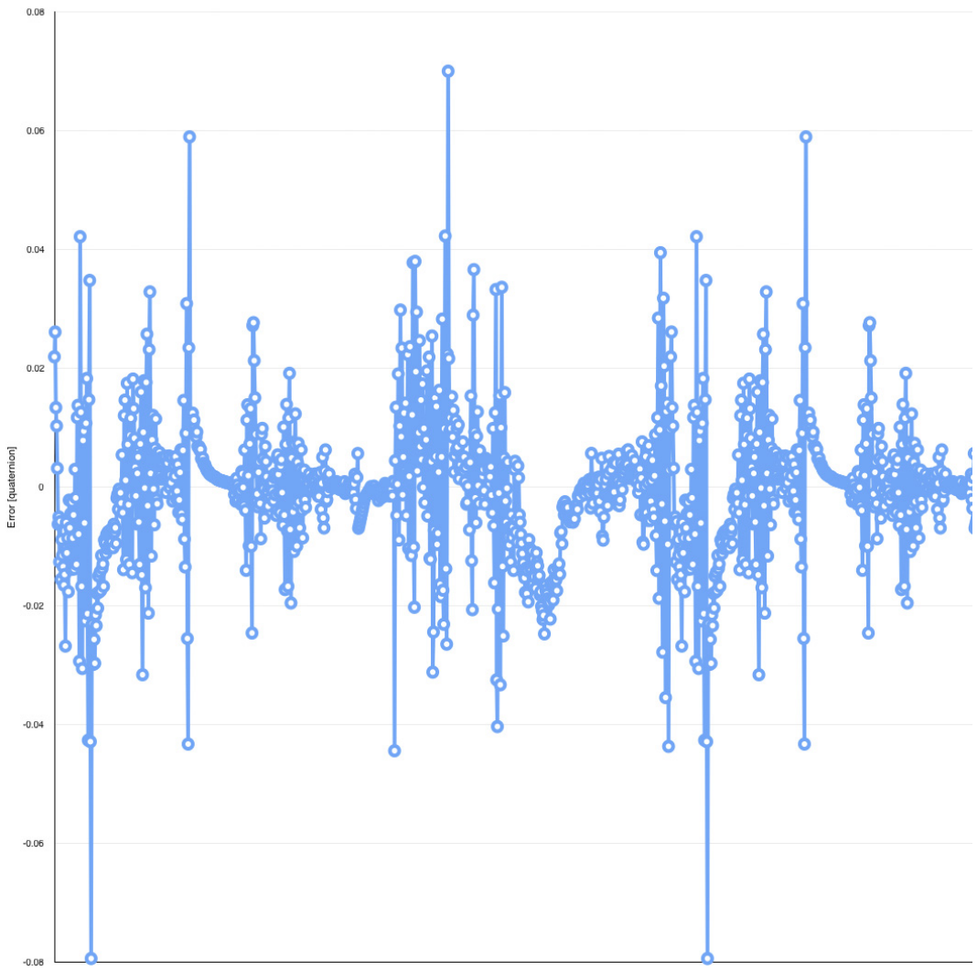


Figure 7.187: Uniform Movement Base Position error, Kd = 450

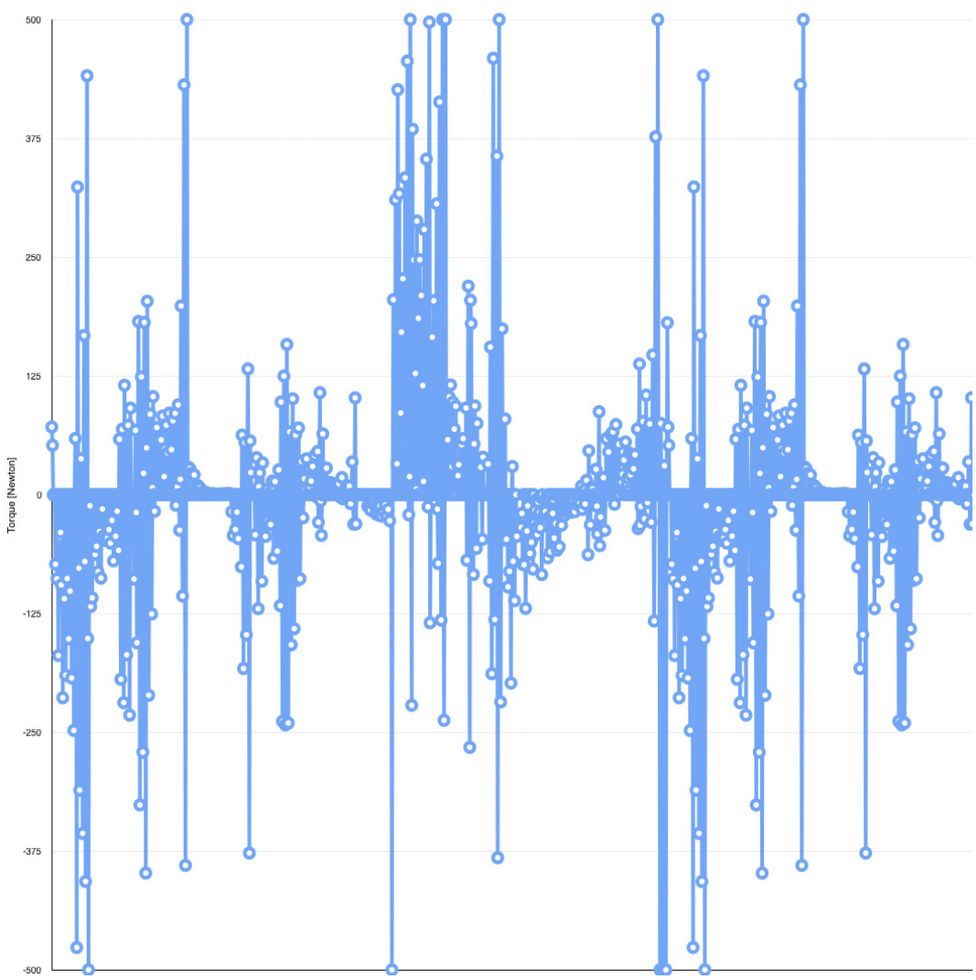


Figure 7.188: Uniform Movement Base Torque, Kd = 450

Bibliography

[1] Thomas Geijtenbeek, *Animating Virtual Characters using Physics-Based Simulation*, PhD Thesis, Utrecht University, 2013.

[2] Thomas Geijtenbeek, Antonie J. van den Bogert, Ben J.H. van Basten, Arjan Egges, *Evaluating the Physical Realism of Character Animations Using Musculoskeletal Models*, Motion in Games: Third International Conference, 2010.

[3] David Baraff, Andrew Witkin, *Physically Based Modeling: Principles and Practice*, Online Siggraph '97 Course notes, 1997.

[4] C. Karen Liu, Aaron Hertzmann, Zoran popovíc, *Learning Physics-Based Motion Style with Nonlinear Inverse Optimization*, SIGGRAPH '05 ACM SIGGRAPH 2005 Papers, 1071-1081, 2005.

[5] Yeuhi Abe, C. Karen Liu, Zoran Popovíc, *Momentum-based Parameterization of Dynamic Character Motion*, Eurographics/ACM SIGGRAPH Symposium on Computer Animation, 2004.

[6] C. Karen Liu, Aaron Hertzmann, Zoran popovíc, *Learning Physics-Based Motion Style with Nonlinear Inverse Optimization*, SIGGRAPH '05 ACM SIGGRAPH 2005 Papers, 1071-1081, 2005.

[7] C. Karen Liu, Zoran popovíc, *Synthesis of Complex Dynamic Character Motion from Simple Animations*, ACM Transactions on Graphics SIGGRAPH, 2002.

[8] Petros Faloutsos, Michiel van de Panne, Demetri Terzopoulos, *Composition of Complex Optimal Multi-Character Motions*, Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, 2006.

[9] Sergey Leviney, Jovan Popovíc, *Physically Plausible Simulation for Character Animation*, Eurographics/ ACM SIGGRAPH Symposium on Computer Animation, 2012.

[10] Keith Grochow, Steven L. Martin, Aaron Hertzmann, Zoran Popovíc, *Style-Based Inverse Kinematics*, ACM Trans. on Graphics SIGGRAPH, 2004.

[11] Ioannis A. Kakadiaris, *Physics-Based Modeling, Analysis and Animation*, Technical Reports (CIS). Paper 274, 1993.

[12] Arthur D. Kuo, Felix E Zajac, *Human Standing Posture: Multijoint Movement Strategies based on Biometrical Constraints*, Progress in Brain Research, Vol. 97, 1993.

[13] Michael Meredith, Steve Maddock, *Real-Time Inverse Kinematics: The Return of the Jacobian*, Tech. Rep. CS-04-06, Dept. of Computer Science, University of Sheffield, pp. 1-15, 2004.

[14] David Tzu-Wei Chen, *Pump It Up: Computer Animation of a Biomechanically Based Model of Muscle using the Finite Element Method*, SIGGRAPH '92 Proceedings of the 19th annual conference on Computer graphics and interactive techniques, 89-98, 1992.

[15] Roy Featherstone, *Rigid Body Dynamics Algorithms*, Rigid Body Dynamics Algorithms, 2008.

[16] Petros Faloutsos, Michiel van de Panne, Demetri Terzopoulos, *Composable Controllers for Physics-Based Character Animation*, SIGGRAPH 28th annual conference on Computer graphics and interactive techniques, 251-260, 2001.

[17] Thomas R. Kurfess, *Robotics and Automation Handbook*, CRC Press, 2004.

[18] Bullet Physics, *Bullet Physics Documentation and Reference Manual*, http://bulletphysics.org/wordpress/, 2015.

[19] J. Martínez, J.C. Nebel, D. Makris, C. Orrite, *Tracking Human Body Parts Using Particle Filters Constrained by Human Biomechanics*, 2008.

[20] Victor B. Zordan, Jessica K. Hodgins, *Motion capture-driven simulations that hit and react*, proceedings of the 2002 ACM SIGGRAPH/Eurographics symposium on Computer animation, 89-96, 2002.

[21] Thomas Geijtenbeek, Nicolas Pronost, Frank van der Stappen, *Simple Data-Driven Control for Simulated Bipeds*, proceedings of the Eurographics/ACM SIGGRAPH Symposium on Computer Animation, 211-219, 2002.

[22] Petros Faloutsos, *Physics-Based Animation and Control of Flexible Characters*, 1995.

[23] Masaki Oshita, Akifumi Makinouchi, *A Dynamic Motion Control Technique for Human-like Articulated Figures*, EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne, Volume 20, 2001.

[24] Sukyung Park, Fay B. Horak, Arthur D. Kuo, *Postural Feedback Responses Scale with Biomechanical Constraints in Human Standing*, Exp Brain Res, 417–427, 2004.

[25] Jehee Lee, Sung Yong Shin, *A Hierarchical Approach to Interactive Motion Editing for Human-like Figures*, SIGGRAPH '99 Proceedings of the 26th annual conference on Computer graphics and interactive techniques, 39-48, 1999.

[26] Charles F. Rose, III Peter-Pike J. Sloan Michael F. Cohen, *Artist-Directed Inverse-Kinematics Using Radial Basis Function Interpolation*, EUROGRAPHICS 2001 / A. Chalmers and T.-M. Rhyne, Volume 20, 2001.

[27] Hyun Joon Shin, Lucas Kovar, Michael Gleicher, *Physical Touch-Up of Human Motions*, Computer Graphics and Applications, 2003. Proceedings. 11th Pacific Conference, 194 - 203, 2003.

[28] Evangelos Kokkevis, *Practical Physics for Articulated Characters*, Game Developers Conference, Sony Computer Entertainment America Research and Development, 2004.

[29] Zoran Popovíc, Andrew Witkin, *Physically Based Motion Transformation*, SIGGRAPH 99, 1999.

[30] Thomas Geijtenbeek, Michiel van de Panne, A. Frank van der Stappen, *Flexible Muscle-Based Locomotion for Bipedal Creatures*, ACM Transactions on Graphics, Vol. 32, Nr. 6 (Proc. of SIGGRAPH Asia 2013), 2013.

[31] Andrew Witkin, Michael Kass, *Spacetime Constraints*, SIGGRAPH '88 Proceedings of the 15th annual conference on Computer graphics and interactive techniques, 159-168, 1998.

[32] Philip Lee, Susanna Wei, Jianmin Zhao, Norman I. Badler, *Strength Guided Motion*, Technical Reports (CIS), Paper 536, 1990.

[33] Nancy S. Pollard, Paul S. A. Reitsma, *Animation of Humanlike Characters: Dynamic Motion Filtering with a Physically Plausible Contact Model*, Proc. of Yale Workshop on Adaptive and Learning Systems, 2001.

[34] Wayne L. Wooten, *Simulation of Leaping, Tumbling, Landing and Balancing Humans*, Doctoral Dissertation, 1998.

[35] Eric Stoneking, *Implementation of Kane's Method for a Spacecraft Composed of Multiple Rigid Bodies*, AIAA Guidance, Navigation, and Control (GNC) Conference, 2013.

[36] Richard T. de Luca, *A Brief Synopsis of Kane's Method*, Paper submitted to the Carnegie Mellon University.

[37] Thomas R. Kane, Peter W. Likins, David A. Levinson, *Spacecraft Dynamics*.

[38] Carlos M. Roithmayr, Abdulrahman H. Bajodah, Dewey H. Hodges and Ye-Hwa Chen, *Corrigendum: New Form of Kane's Equations of Motion for Constrained Systems*, NASA, 2007.

[39] Balafoutis, Constantinos A., Patel, Rajnikant V., *Dynamic Analysis of Robot Manipulators, The Springer International Series in Engineering and Computer Science*, 1991.

[40] R. Brent Gillespie, *Kane's Equations for Haptic Display of Multibody Systems*, Haptics-e: The Electronic Journal of Haptics Research, Vol.3 No.2, 2013.

[41] Kenny Erleben, Jon Sporring, Knud Henriksen, Henrik Dohlmann, *Physics-based Animation*, Charles River Media, Graphics Series, 2005.

[42] Constantinos A. Balafoutis, Rajnikant V. Patel, *Dynamic Analysis of Robot Manipulators: A Cartesian Tensor Approach*, Springer Science and Business Media, 2012.

[43] M.H.P. Dekker, *Zero-Moment Point Method for Stable Biped Walking*, Internship report Eindhoven, 2009.

[44] Van C. Mow, Rik Huiskes, *Basic Orthopaedic Biomechanics and Mechano-biology*, Lippincott Williams and Wilkins, 2005.

[45] Johannes Gerstmayr, *HOTINT – A C++ Environment for the Simulation of Multibody Dynamics Systems and Finite Elements*, ECCOMAS Thematic Conference, 2009.

[46] O. Verlinden, G. Kouroussis, S. DatoussaÄd, C. Conti, *Open source symbolic and numerical tools for the simulation of multibody systems*, ECCOMAS Thematic Conference, 2005.

[47] Adriano Macchietto, Victor Zordan, Christian R. Shelton, *Momentum Control for Balance*, ACM Transactions of Graphics 28, 80:1-80:8, 2009.

[48] Kiven Wampler, Zoran Popóvic, *Optimal Gait and Form for Animal Locomotion*, SIGGRAPH 2009.